

FUJITSU Software

PRIMECLUSTER

A horizontal decorative band with a red-to-dark-red gradient. It features abstract, glowing white and red lines that swirl and curve across the band, creating a sense of motion and energy.

Reliant Monitor Services (RMS) with Wizard Tools Configuration and Administration Guide 4.5

Oracle Solaris / Linux

J2S2-1687-02ENZ0(04)
June 2019

Copyright and Trademarks

UNIX is a registered trademark of The Open Group in the United States and other countries.

Red Hat is a trademark of Red Hat, Inc. in the United States and other countries.

Linux is a trademark or registered trademark of Mr. Linus Torvalds in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

All other hardware and software names used are trademarks of their respective companies.

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Requests

- No part of this documentation may be reproduced or copied without permission of FUJITSU LIMITED.
- The contents of this documentation may be revised without prior notice.

All Rights Reserved, Copyright (C) FUJITSU LIMITED 2017-2019.

Contents

1	Preface	1
1.1	About this manual	1
1.2	PRIMECLUSTER documentation list	2
1.3	Conventions	3
1.3.1	Notation	3
1.3.1.1	Prompts	4
1.3.1.2	Manual page section numbers	4
1.3.1.3	The keyboard	4
1.3.1.4	Typefaces	4
1.3.1.5	Example 1	5
1.3.1.6	Example 2	5
1.3.2	Command line syntax	5
1.4	Important notes and cautions	6
1.5	Abbreviations	7
1.6	Revision history	7
2	Introduction	9
2.1	PRIMECLUSTER overview	9
2.2	How RMS provides high availability	11
2.2.1	Applications, resources, and objects	11
2.2.2	Relationship of RMS configurations to the real world	14
2.2.3	Node and application failover	16
2.2.4	Controlled applications and controller objects	16
2.2.4.1	Follow controllers	18
2.2.4.2	Scalable controllers	19
2.2.4.3	Further notes about controllers	21
2.3	How the Wizard Tools provide easy configuration	21
2.4	RMS wizard products	22
2.4.1	RMS Wizard Tools	24
2.4.2	RMS Wizard Kit	24
2.5	Cluster Admin administration tool	25
2.6	RMS components	25
2.6.1	Base monitor	25
2.6.2	Detectors and states	26
2.6.3	Scripts	26
2.6.4	RMS CLI	27
2.7	Object types	30
2.8	Object attributes	30
2.9	Environment variables	31
2.9.1	Setting environment variables	31
2.9.2	Script execution environment variables	32

Contents

2.10	RMS Directory structure	33
3	Advanced RMS concepts	35
3.1	State transition	35
3.1.1	Internal organization	35
3.1.1.1	Application and resource description	36
3.1.1.2	Messages	36
3.1.2	Initializing	37
3.1.3	Online processing	41
3.1.3.1	Online request	41
3.1.3.2	PreCheckScript	43
3.1.3.3	Online processing in a logical graph of a userApplication	44
3.1.3.4	Unexpected reports during online processing	46
3.1.3.5	Fault situations during online processing	46
3.1.3.6	Initialization when an application is already online	47
3.1.4	Offline processing	48
3.1.4.1	Offline request	48
3.1.4.2	Offline processing in a logical graph of a userApplication	49
3.1.4.3	Unexpected reports during offline processing	50
3.1.4.4	Fault situations during offline processing	51
3.1.4.5	Object is already in Offline state	51
3.1.4.6	Object cannot be sent to Offline state	51
3.1.5	Fault processing	52
3.1.5.1	Faults in the online state or request processing	52
3.1.5.2	Offline faults	54
3.1.5.3	AutoRecover attribute	55
3.1.5.4	Fault during offline processing	55
3.1.5.5	Examples of fault processing	55
3.1.5.6	Fault clearing	58
3.1.5.7	SysNode faults	59
3.1.6	Switch processing	60
3.1.6.1	Switch request	60
3.1.7	Special states	62
3.1.7.1	Restrictions during maintenance mode	62
3.1.7.2	The Inconsistent state	64
3.2	RMS heartbeat operation	65
3.2.1	Operational details	67
3.2.1.1	ELM lock management	67
3.2.1.2	First node startup	68
3.2.1.3	Second node startup	68
3.2.1.4	Third and subsequent nodes	69
3.2.1.5	Node or RMS down scenario	69
3.2.1.6	Slow response scenario	70
3.2.2	Default initialization in hvenv	71

3.2.3	Manually controlling ELM in hvenv.local	71
3.3	Scalable controllers	72
3.3.1	Controller overview	72
3.3.2	Scalable controllers and applications	72
3.3.2.1	Scalable applications	73
3.3.2.2	Benefits of scalable controllers	73
3.3.2.3	Attributes for scalable controllers	74
3.3.3	Online/offline processing and state transitions	75
3.3.3.1	How controller states depend on controlled application states	75
3.3.3.2	About nodes of which application become the online state	76
3.3.3.3	Request propagation to controlled applications	76
3.3.3.4	Controller warning state	77
3.3.3.5	Application warning state	77
3.3.3.6	Controller state change script	77
3.3.3.7	Sequenced online/standby/offline and application groups	80
3.3.3.8	Auto Startup on a sub-cluster	81
3.3.3.9	Switchover on a sub-cluster	82
3.3.3.10	Manual switching of child applications	82
3.3.3.11	Operation during maintenance mode	82
4	Using the Wizard Tools interface (hvw)	89
4.1	Overview	89
4.1.1	RMS Wizard types	90
4.1.1.1	Turnkey wizards	91
4.1.1.2	Resource wizards	91
4.2	General configuration procedure	91
4.3	Creating and editing a configuration	92
4.3.1	Using the wizard menus	93
4.3.2	Main configuration menu	94
4.3.2.1	Main configuration menu when RMS is not active	94
4.3.2.2	Main configuration menu when RMS is running	98
4.3.3	Secondary menus	99
4.3.4	Basic and non-basic settings	100
4.4	Activating a configuration	102
4.5	Configuration elements	105
4.5.1	Scripts	105
4.5.2	Detectors	105
4.5.3	RMS objects	106
5	Example of configuration change	107
5.1	Setting PreCheckScript in cluster application	107
5.2	Changing timeout period of controller	113
6	Using the Cluster Admin GUI	121

Contents

6.1	Overview	121
6.2	Starting the Cluster Admin GUI	122
6.2.1	Web-Based Admin View	122
6.2.2	Login	123
6.2.3	Main Cluster Admin window	125
6.2.4	Cluster Admin message view	127
6.3	Monitoring RMS with Cluster Admin	128
6.3.1	RMS tree	129
6.3.2	Pop-up context menus	131
6.3.3	Pop-up confirmation dialogs	133
6.3.4	Displaying environment variables	133
6.3.5	Displaying object states	135
6.3.6	Configuration information or object attributes	137
7	Additional administrative tools	139
7.1	Using the RMS clusterwide table	139
7.1.1	Using context menus from the clusterwide table	142
7.2	Using RMS graphs	143
7.2.1	RMS full graph	143
7.2.2	Application graph	148
7.2.3	Subapplication graph	149
7.2.4	Composite subapplication graph	151
7.2.5	Using pop-up context menus from the graph	153
7.2.6	Changing the displayed detail level	154
7.3	Interpreting display changes	158
7.3.1	Display during RMS configuration changes	158
7.3.2	Display after RMS shutdown	159
7.4	Viewing RMS log messages	162
7.4.1	Common procedures for switchlog and application log	166
7.4.2	Time filter	167
7.4.3	Keyword filters	168
7.4.3.1	Resource Name	168
7.4.3.2	Severity	169
7.4.3.3	Non-zero exit code	170
7.4.3.4	Keyword	170
7.4.4	Text search	171
7.4.5	Removing filters	171
8	Controlling RMS operation	173
8.1	Managing RMS nodes	173
8.1.1	Starting RMS	173
8.1.2	Starting RMS automatically at boot time	179
8.1.3	Stopping RMS	181
8.1.4	Clearing a SysNode Wait state	186

8.2	Managing RMS applications	188
8.2.1	Overriding automatic application startup	188
8.2.2	Switching an application	190
8.2.3	Starting an application	193
8.2.4	Stopping an application	194
8.2.5	Resetting an application	195
8.2.6	Clearing a fault	196
8.2.7	Activating an application	198
8.3	Managing resources	199
8.3.1	Switching a resource	199
8.3.2	Starting a resource	201
8.3.3	Stopping a resource	202
8.3.4	Notes on starting and stopping a resource	204
8.3.5	Fault Traces of Resources	204
8.4	Using maintenance mode	208
8.4.1	Entering maintenance mode	208
8.4.2	Maintenance mode operating notes	211
8.4.2.1	Overall cluster restrictions in maintenance mode	211
8.4.3	Exiting maintenance mode	211
8.4.4	Maintenance mode CLI: hvutil -m and -M	213
9	Appendix—Site preparation	215
9.1	Network database files	216
9.1.1	/etc/hosts	216
9.1.1.1	Network interface names in /etc/hosts	217
9.1.2	/root/.rhosts (Linux) and /.rhosts (Solaris)	217
9.2	Configuration resource definitions	219
9.2.1	/opt/SMAW/SMAWRrms/etc/hvipalias	219
9.2.2	/opt/SMAW/SMAWRrms/etc/hvconsoles	220
9.3	Linux file systems	221
9.3.1	/etc/fstab.pcl	221
9.3.1.1	Configuring file systems for particular applications	222
9.3.1.2	Clusterwide configuration issues	222
9.4	Solaris file systems	223
9.4.1	NFS Lock Failover	224
9.5	NFS servers	225
9.6	Log files	225
9.6.1	/var/log/messages (Linux) or /var/adm/messages (Solaris)	225
9.7	Other system services and databases	226
10	Appendix—States	227
10.1	Basic states	227
10.2	State details	229

Contents

11	Appendix—Object types	231
12	Appendix—Attributes	233
12.1	Attributes available to the user	233
12.2	Attributes managed by configuration wizards	240
13	Appendix—Environment variables	245
13.1	Setting environment variables	245
13.2	Global environment variables	246
13.3	Local environment variables	252
13.4	Script execution environment variables	256
14	Appendix—Troubleshooting	259
14.1	Overview	260
14.2	Debug and error messages	261
14.3	Log file categories	262
14.4	Managing base monitor log output	264
14.4.1	Managing base monitor log levels	264
14.4.2	Controlling base monitor output to the system log	266
14.5	Managing application log output	266
14.5.1	Controlling debug messages from standard scripts	267
14.6	Managing detector log output	268
14.6.1	Debug messages and log levels	268
14.6.2	Setting the detector log level	269
14.6.2.1	Setting the detector log level with hvutil -L	269
14.6.2.2	Setting the log level in the Wizard Tools	270
14.7	Interpreting RMS log messages	271
14.7.1	switchlog message format	271
14.7.2	Application log message format	273
14.7.3	Program log message format	274
14.8	RMS log file cleanup	274
14.8.1	hvmlogclean	275
14.8.2	hvmlogcontrol	275
14.8.3	Logging of background scripts	275
14.9	RMS troubleshooting	276
14.10	Collecting information for advanced troubleshooting	280
14.10.1	Using the hvdump command (RMS)	280
14.11	SNMP Notification of Resource Failure	280
15	Appendix—RMS command line interface	283
15.1	Available RMS CLI commands	283
16	Appendix—Release information	287
Glossary		289

Abbreviations 305

Figures 309

Index 317

Tables 351

Contents

1 Preface

Reliant[®] Monitor Services (RMS) is a software monitor designed to guarantee the high availability of applications in a cluster of nodes. This manual describes how to configure RMS using the RMS Wizards and how to administer RMS using the Cluster Admin GUI.

This manual introduces RMS components and describes necessary configuration data and management tasks for operating RMS correctly. This manual also describes diagnostic procedures, including how to view and interpret RMS log files.

This manual is aimed at system administrators who create and maintain RMS configurations. Familiarity with the following system functions and components is assumed:

- PRIMECLUSTER family of products
- Oracle Solaris or Linux operating system
- Non-PRIMECLUSTER products such as volume managers and storage area networks.

This document assumes that the PRIMECLUSTER software has been installed as described in "Software Release Guide" and "PRIMECLUSTER Installation Guide" for your operating system.

1.1 About this manual

This manual is structured as follows:

- The "Introduction" provides an introduction to RMS terminology and describes basic principles of operation.
- The "Advanced RMS concepts" provides background details about RMS operation including state detection and transition processing.
- The "Using the Wizard Tools interface (hvw)" describes how to configure RMS using the RMS Wizard Tools.
- The "Example of configuration change" illustrates the procedure on how to change the configuration of cluster application using RMS Wizard Tools.
- The "Using the Cluster Admin GUI" describes how to start and use the Cluster Admin graphical user interface (GUI), which is the primary tool for all RMS administrative functions.

- The "Additional administrative tools" describes the RMS clusterwide table, RMS graphs, and the RMS log viewer.
- The "Controlling RMS operation" describes common RMS administrative functions available through the Cluster Admin GUI, including the equivalent CLI procedures.
- The "Appendix—Site preparation" describes network and file settings required for RMS operation.
- The "Appendix—States" lists the object states that are supported by RMS.
- The "Appendix—Object types" lists the object types that are supplied with RMS.
- The "Appendix—Attributes" lists the attributes that are supported by RMS object types.
- The "Appendix—Environment variables" describes the RMS environment variables.
- The "Appendix—Troubleshooting" describes how to troubleshoot RMS using graphical user interface (GUI) and command line interface (CLI) tools.
- The "Appendix—RMS command line interface" lists the RMS administrative CLI commands.

1.2 PRIMECLUSTER documentation list

The documents listed below provide details about PRIMECLUSTER products. Please contact your sales representative for ordering information.

- "PRIMECLUSTER Concepts Guide (Oracle Solaris, Linux)"—Provides conceptual details on the PRIMECLUSTER family of products.
- "PRIMECLUSTER Installation and Administration Guide"—Provides instructions for installing and upgrading PRIMECLUSTER products.
- "PRIMECLUSTER Web-Based Admin View Operation Guide"—Provides information on using the Web-Based Admin View management GUI.
- "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide"—Provides instructions for configuring and administering the PRIMECLUSTER Cluster Foundation.

- "PRIMECLUSTER Global Disk Services Configuration and Administration Guide"—Provides information on configuring and administering Global Disk Services (hereinafter GDS).
- "PRIMECLUSTER Global File Services Configuration and Administration Guide"—Provides information on configuring and administering Global File Services (hereinafter GFS).
- "PRIMECLUSTER Global Link Services Configuration and Administration Guide: Redundant Line Control Function"—Provides information on configuring and administering the redundant line control function for Global Link Services (hereinafter GLS).
- "PRIMECLUSTER Global Link Services Configuration and Administration Guide: Redundant Line Control Function for Virtual NIC Mode"—Provides information on configuring and administering the Virtual NIC Mode for Global Link Services (GLS).
- "PRIMECLUSTER Global Link Services Configuration and Administration Guide: Multipath Function"—Provides information on configuring and administering the multipath function for Global Link Services (GLS).
- "PRIMECLUSTER DR/PCI Hot Plug User's Guide"—Provides the operation of Dynamic Reconfiguration function and PCI Hot Plug function by using PRIMECLUSTER.
- "PRIMECLUSTER Messages"—Provides how to set the environment for PRIMECLUSTER. Messages output during system operation are also provided.
- "Software Release Guide" and "Installation Guide"—Provide late-breaking information about installation, configuration, and operation about PRIMECLUSTER.

1.3 Conventions

To standardize the presentation of material, this manual uses a number of notational, typographical, and syntactical conventions.

1.3.1 Notation

This manual uses the following notational conventions.

1.3.1.1 Prompts

Command line examples that require system administrator (or root) rights to execute are preceded by the system administrator prompt, the hash sign (#). Entries that do not require system administrator rights are preceded by a dollar sign (\$).

In some examples, the notation `<nodename>#` indicates a root prompt on the specified node. For example, a command preceded by `shasta1#` would mean that the command was run as user `root` on the node named `shasta1`.

1.3.1.2 Manual page section numbers

The command names and file names are followed by their manual page section numbers in parentheses, *e.g.*, `cp(1)`.

Section numbers of Oracle Solaris 11 or earlier are described in this manual. Since some of the section numbers are updated in Oracle Solaris 12, read and follow the table below:

Section no. of Solaris 11 or earlier	Section no. of Solaris 12	Description
1M	8	System maintenance and Management command
4	5	File format
5	7	Others
7	4	Special device file

1.3.1.3 The keyboard

Keystrokes that represent nonprintable characters are displayed as key icons such as `[Enter]` or `[F1]`. For example, `[Enter]` means press the key labeled **Enter**; `[Ctrl-b]` means hold down the key labeled **Ctrl** or **Control** and then press the `[B]` key.

1.3.1.4 Typefaces

The following typefaces highlight specific elements in this manual.

Typeface	Usage
Constant Width	Computer output and program listings; commands, file names, and other literal programming elements in the main body of text.
<i>Italic</i>	Variables in a command line that you must replace with an actual value. May be enclosed in angle brackets to emphasize the difference from adjacent text; for example, <code><nodename>RMS</code> ; unless directed otherwise, you should not enter the angle brackets. The name of an item in a character-based or graphical user interface. This may refer to a menu item, a radio button, a checkbox, a text input box, a panel, or a window title.
Bold	Items in a command line that you must type exactly as shown.

Typeface conventions are shown in the following examples.

1.3.1.5 Example 1

Several entries from an `/etc/passwd` file are shown below:

```
root:x:0:1:0000-Admin(0000):/:/sbin/ksh
sysadm:x:0:0:System Admin.:/usr/admin:/usr/sbin/sysadm
setup:x:0:0:System Setup:/usr/admin:/usr/sbin/setup
daemon:x:1:1:0000-Admin(0000):/:
```

1.3.1.6 Example 2

To use the `cat(1)` command to display the contents of a file, enter the following command line:

```
$ cat file
```

1.3.2 Command line syntax

The command line syntax observes the following conventions.

Symbol	Name	Meaning
[]	Brackets	Enclose an optional item.
{ }	Braces	Enclose two or more items of which only one is used. The items are separated from each other by a vertical bar ().
	Vertical bar	When enclosed in braces, it separates items of which only one is used. When not enclosed in braces, it is a literal element indicating that the output of one program is piped to the input of another.
()	Parentheses	Enclose items that must be grouped together when repeated.
...	Ellipsis	Signifies an item that may be repeated. If a group of items can be repeated, the group is enclosed in parentheses.

1.4 Important notes and cautions



Important

Indicates important information.



Caution

Indicates a situation that can cause harm to data.



Note

Indicates information that needs special attention.

1.5 Abbreviations

Oracle Solaris might be described as Solaris, Solaris Operating System, or Solaris OS.

If "Solaris X" is indicated in the reference manual name of the Oracle Solaris manual, replace "Solaris X" with "Oracle Solaris 10 (Solaris 10)," "Oracle Solaris 11 (Solaris 11)," or "Oracle Solaris 12 (Solaris 12)."

Red Hat Enterprise Linux is abbreviated as RHEL.

RHEL is abbreviated as Linux.

PRIMEQUEST 3000/2000 Series are abbreviated as PRIMEQUEST.

1.6 Revision history

Revision	Location	Manual code
Deleted the descriptions of NFS.	9.3	J2S2-1687-02ENZ0(01)
Changed the description of MonitorOnly attribute.	12.1	
Changed the description of RELIANT_SHUT_MIN_WAIT.	13.2	J2S2-1687-02ENZ0(02)
Changed the descriptions of configuration scripts.	2.6.3	J2S2-1687-02ENZ0(03)
Changed the possible values of the RMS script execution environment variable HV_SCRIPT_TYPE.	13.4	
Changed the description of StateChangeScript.	2.6.3	J2S2-1687-02ENZ0(04)

2 Introduction

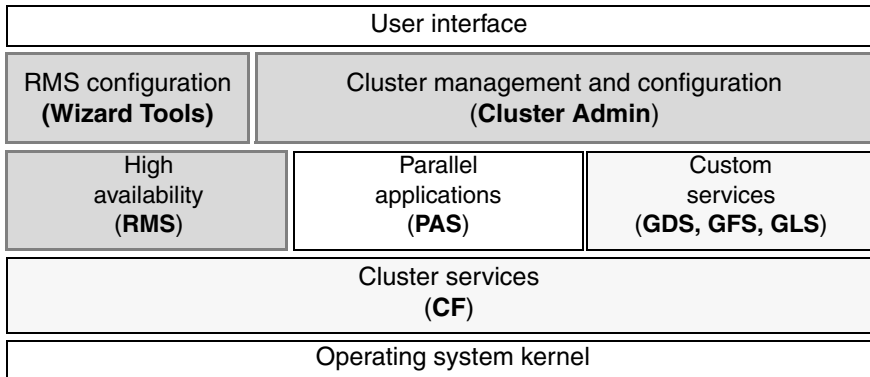
This chapter contains general information on Reliant Monitor Services (RMS), introduces the PRIMECLUSTER family of products, details how RMS, RMS Wizard Tools, and the RMS Wizard Kit work together to produce high-availability configurations, and introduces Cluster Admin.

Chapter contents:

- "PRIMECLUSTER overview"
- "How RMS provides high availability"
- "How the Wizard Tools provide easy configuration"
- "RMS wizard products"
- "Cluster Admin administration tool"
- "RMS components"
- "RMS CLI"
- "Object types"
- "Object attributes"
- "Environment variables"
- "RMS Directory structure"

2.1 PRIMECLUSTER overview

The PRIMECLUSTER family of products is an integrated set of cluster services, including configuration and administration services, high availability, scalability, parallel application support, cluster file system, and cluster volume management. Figure 1 illustrates the relationship of PRIMECLUSTER services to each other and to the operating system environment.



GDS: Global Disk Services
 GLS: Global Link Services
 GFS: Global File Services

Figure 1: Overview of PRIMECLUSTER products

This manual focuses on PRIMECLUSTER products and services that relate to high availability operation (shown with a solid gray background in the figure above). They are as follows:

- **RMS**—This high availability manager is a software monitor that provides high availability (HA) for customer applications in a cluster of nodes. Its task is to monitor systems and application resources, to identify any failures, and to provide application availability virtually without interruption in the event of any such failures.

RMS also provides integrated services for market-specific applications. Contact field engineers for availability and details.

- **RMS Wizard Tools**—This configuration tool provides a character-based interface to create RMS configurations. It includes templates for generic applications and commonly used resources.

The RMS Wizard Kit works with the Wizard Tools to configure popular enterprise products for operation with RMS.

- **Cluster Admin**—The Cluster Admin GUI is the primary administrative tool for RMS.
- **Cluster Foundation (CF)**—This comprehensive suite provides cluster administration and communication services for user applications and other PRIMECLUSTER services.

PRIMECLUSTER products shown with a dotted background in the figure above are described in their respective manuals. See the section "PRIMECLUSTER documentation list" in the Preface.

2.2 How RMS provides high availability

RMS provides high availability of a customer's application by controlling and monitoring the state of all resources in use by a given application. Resources include items such as network interfaces, local and remote file systems, and storage area networks.

2.2.1 Applications, resources, and objects

RMS relies on a virtual representation of the cluster called a **configuration**. The configuration represents each machine, application, and system resource as an object, and the objects are logically arranged in a tree structure according to their dependencies. For instance, suppose a user application depends on a network interface and a file system in order to operate properly. In the tree structure, the corresponding application object would appear as a parent and the network and file system objects would appear as its children. The tree structure is commonly known as a **graph**.

Each object in the graph contains the **state** of the corresponding item along with any other parameters that may be required. An object is typically in the **online** (enabled, available) state or the **offline** (disabled, unavailable) state, but other states are possible according to the type of object. For the complete list of states supported by RMS, see "Appendix—States".

At runtime, the configuration is managed by the **RMS base monitor**, which initiates actions when an object's state changes, or, in the case of a **timeout**, when an object has remained in the same state for some specified time interval even though a change was expected. This design is known as a **state machine**.

Nodes and heartbeats

Machines that are members of a cluster are called nodes. When RMS monitors the health of a node, its highest priority is to detect a complete failure of the node or its base monitor. Its second priority is to detect slow response times that may be caused by system overloads. RMS uses two mechanisms, heartbeat and ELM to detect these problems.

Refer to the "RMS heartbeat operation".

Detectors

RMS monitors each resource by using **detectors**, which are processes that deliver status reports to the RMS base monitor process. RMS interprets the status reports to determine the state of the corresponding virtual object. When an object's state changes, RMS takes action according to the parameters set in the object. Each object may be associated with a detector.

Detectors are persistent: when RMS starts on a cluster node it starts the detectors for its configuration, which normally continue to run on that node until RMS is shut down. RMS has the ability to restart a detector if it terminates prematurely.

A complete list of the states that can be reported by detectors or displayed in the user interface is presented later in this chapter.

Note that RMS does not use detector processes to monitor the state of machines in the cluster. Instead, each node transmits a **heartbeat** signal at regular intervals, and RMS uses this to determine machine states and connectivity.

Scripts

Each object type has an associated set of **scripts**. A script is a command string (possibly including pipes, redirection, command interpolation, and variable substitution) that can be executed by the operating system shell—in other words, a valid shell script. Normally, each script is designed to interact with items in the operating system such as user applications or physical resources. Scripts provide the only means for RMS to directly influence items outside its virtual representation.

Some scripts are reactive: they define the actions that RMS should take in response to state changes. Other scripts are proactive: they define the actions that RMS should use to take control of individual objects. For instance, RMS would process one script when a resource reports a transition from the online state to the offline state; however, RMS would process a different script when it must force the resource to the offline state.

Scripts are transient: after performing their programmed tasks, they exit and return a status code to the base monitor.

A complete list of the scripts that may be specified for RMS objects is presented later in this chapter.

Object types

Most high-availability applications rely on a set of physical resources such as network interfaces, files systems, or virtual disks. RMS represents these as `gResource` objects. Most `gResource` objects have scripts that allow them to be brought online or taken offline.

Internally, RMS represents an actual application that runs in the operating system environment as a `userApplication` object. The set of `gResource` objects that represent the actual application's resource requirements are called its **dependent resources**. Bringing a `userApplication` object to the online state, along with all of its dependent resources, is called **online processing**. Taking a `userApplication` object to the offline state, along with all of its dependent resources, is called **offline processing**.

Each node that may run one or more applications in the high availability configuration is represented by a `SysNode` object. Like `gResource` objects and `userApplication` objects, `SysNode` objects can be brought online or taken offline, and they have an associated set of scripts. However, booting up or shutting down the corresponding physical machine requires more than simple script processing.

For the complete list of the RMS object types supported by the Wizard Tools, see "Appendix—Object types".

Shutdown Facility

While scripts and detectors provide a direct interface between RMS and the operating system, the **Shutdown Facility** (SF) provides an indirect interface to the machines in the cluster. When it necessary to take a `SysNode` object offline, RMS works with the SF to guarantee that the corresponding node has been physically shut down, or **killed**. RMS waits for successful completion of the node kill before switching any `userApplication` to another `SysNode`. This prevents any user application from running on two machines at the same time, which could lead to data corruption.

For more information about the Shutdown Facility, see "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide" for your operating system.

2.2.2 Relationship of RMS configurations to the real world

It is important to understand that RMS does not interact directly with "real-world" items such as machines, users' applications, or system resources—it interacts only with the objects in its virtual representation. Figure 2 illustrates the relationship between an actual user application in the operating system environment and the corresponding `userApplication` object in an RMS configuration.

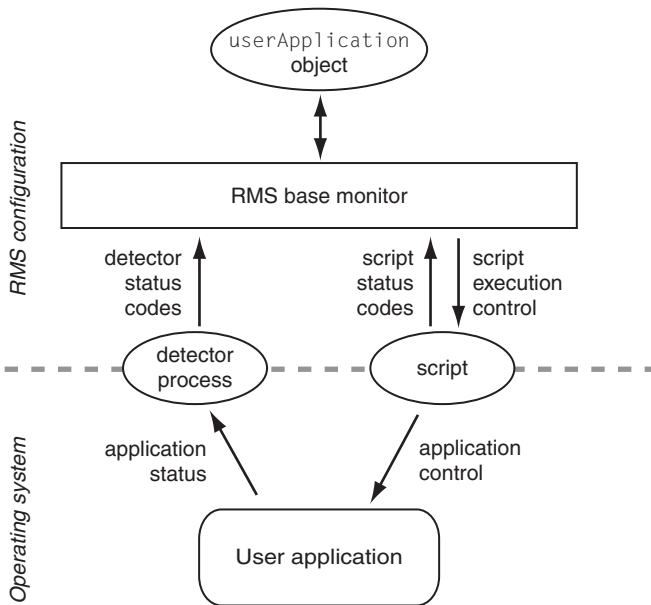


Figure 2: Interface between RMS and the operating system

Note that the interface between the RMS virtual representation and the actual operating system depends entirely on the scripts and detectors provided by the configuration tools. The script in the figure represents any of the standard scripts discussed later in this chapter: it reports whether or not it completed its tasks successfully by returning a status code, and RMS combines this with the status code from the object's detector to determine the object's state. RMS has no other way to determine what actually happened to the user application in the operating system environment (the part of the figure below the dashed line).

For instance, if a `userApplication` object's `Online` script reports success, its detector reports that it is online, and all of its resources are online, then RMS considers that object to be online, regardless of the state of the actual user application. Similarly, if a resource object's detector reports an `Offline` state, it does not necessarily mean that the physical resource is unavailable.



For reliable high availability operation, RMS requires scripts that properly control the corresponding real world items, and detectors that accurately reflect the items' states.

Configuration terminology

This manual discusses configuration procedures within the RMS context (represented by the part of Figure 2 above the dashed line). Strictly speaking, our principle concern is with `SysNode` objects, `userApplication` objects, and other RMS entities, and not the real-world items they represent.

However, it is intuitive to use terms such as "node" instead of "SysNode object" and "application" instead of "userApplication object," because the relationships are so close, and because it is always understood we are working from the RMS perspective. This also helps to simplify many of the technical discussions. Therefore, unless there is a need to distinguish between an RMS object and the actual item it represents, this manual and the configuration tools it describes use the following terms interchangeably:

- "node" and "SysNode object" and "SysNode"
- "application" and "userApplication object" and "userApplication"
- "resource" and "gResource object" and "gResource"

The descriptions of object states and attributes are abbreviated similarly. For instance, instead of "the `gResource` object that monitors the mount point `xyz` is in the `Offline` state," it is customary to say, "the `xyz` file system is offline." It is also common to refer to a script by its attribute name, so "the script specified by the `PreOnlineScript` attribute" becomes simply "the `PreOnlineScript`."

All the objects that are managed by RMS are collectively described as "RMS resource."

2.2.3 Node and application failover

During normal operation, one instance of RMS runs on each node in the cluster. Every instance communicates with the others to coordinate the actions configured for each `userApplication`. If a node crashes or loses contact with the rest of the cluster, then RMS can switch all `userApplication` objects from the failed node to a surviving node in the cluster. This operation is known as **failover**.

Failover can also operate with individual applications. Normally, a `userApplication` object is allowed to be online on only one node at a time. (Exceptions to this rule are shared objects like Oracle RAC `vdisk`.) If a fault occurs within a resource used by a `userApplication` object, then only that `userApplication` can be switched to another node in the cluster. `userApplication` failover involves offline processing for the object on the first node, followed by online processing for the object on a second node.

There are also situations in which RMS requires a node to be shut down, or **killed**. In any case, before switching applications to a new node, RMS works together with the PRIMECLUSTER Shutdown Facility to guarantee that the original node is completely shut down. This helps to protect data integrity.

RMS also has the ability to recover a resource locally; that is, a faulted resource can be brought back to the online state without switching the entire `userApplication` to another cluster node.

2.2.4 Controlled applications and controller objects

In some situations, it is desirable for one application to control another in a parent/child relationship. Consider a scenario in which a bank teller application depends on the local network (represented by a network resource object) and a database application. This can be represented by the graph in Figure 3.

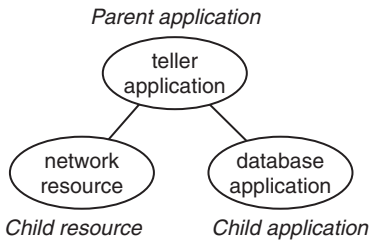


Figure 3: Parent application with two dependencies

Assume that if the network fails in some way, or if the database fails in some way, then the parent teller application cannot complete any transactions. The lines joining the objects in the figure indicate these dependencies. From the RMS perspective, then, we would like both the network resource and the database application to be configured in similar ways: they should both act as dependent resources that must be online if the teller application is to function properly.

However, RMS does not allow any application to be directly configured as the child of another application. Instead, RMS accommodates parent/child relationships between applications by providing an intermediate `controller` object, which is often simply called a **controller**. Unlike other resource objects, a controller has no scripts or detectors. Instead, it propagates online and offline requests from the parent to the child application, and it determines its status from that of the child application.

Figure 4 demonstrates how RMS would represent the banking scenario with the teller application, the controller, and the database application all running on `node1`. For the purposes of this example and the discussions that follow, only the applications and the controller are included in the illustration; the resource object representing the network interface is not shown.

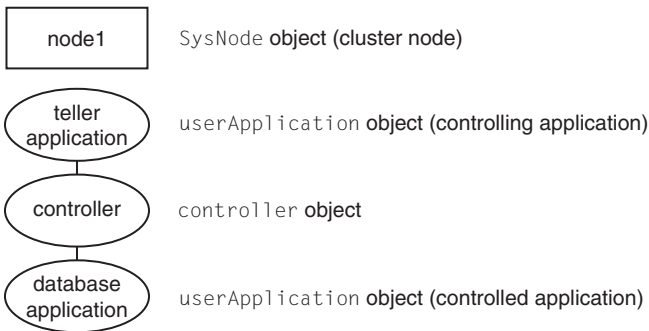


Figure 4: RMS representation of controlled application

i In accordance with the manual of the PRIMECLUSTER Products that support the operation mode of the controller objects, use the control of the applications by the controller objects.

i Each controlled application requires a separate controller as a child of the parent application. Also, controllers exist only for internal RMS management purposes—there is no equivalent in the context of the real-world operating system.

If a child changes to an offline or faulted state, RMS will attempt to recover locally or switch the parent, the child, and the dependent resources to another node.

2.2.4.1 Follow controllers

RMS controllers operate in follow mode: the child application must always run on the same node as the parent. If the parent is switched to another node, the application and all its dependent resources will be switched there too. Likewise, if the child application fails in a way that requires it to be switched to another node, then the parent must be switched there as well.

Assume that the parent teller application tree is originally online on `node1` as shown in Figure 4 above. If either the parent or child application needs to be switched to `node2` for any reason, the rest of the tree follows. Figure 5 illustrates the result.

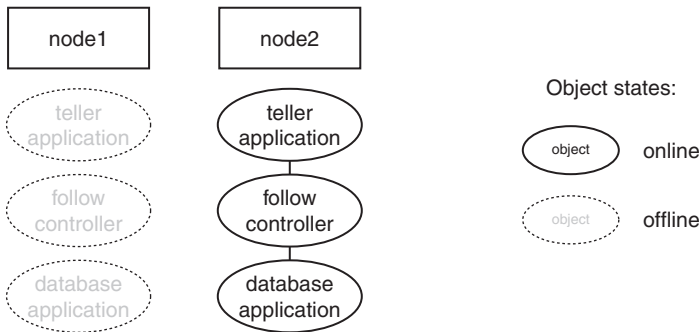


Figure 5: Result of follow-mode switchover

Note the state of the controller in Figure 5. Like the child application, it is brought online only on the same node as the parent. Follow controllers can guarantee that a group of applications and their resources remain closely coupled, so they always run together on the same machine.



When RMS switches an application from `node1` to `node2`, no objects are moved within the corresponding graph. Instead, the objects in the part of the graph corresponding to `node1` are first taken offline, and then the objects in the part of the graph corresponding to `node2` are brought online. The sequence used by RMS in an actual configuration is crucial to high availability operation. For a more detailed discussion, see "Site preparation."

2.2.4.2 Scalable controllers

Generally speaking, **scalability** is the ability of the overall system to adapt to changes in resources or workload. One of the main features of RMS scalable controllers is that they allow the parent and child applications to run on separate machines. This not only provides more flexibility but it may also prevent delays or outages when resources fail in certain combinations.

In our banking scenario, for example, the teller application depends on a network and a database application. Suppose the file system on `node1` fails and the database goes offline. If the database controller is operating in follow mode, RMS will attempt to switch the teller and database to `node2`. However, if the network on `node2` is offline or faulted, the teller can't be brought online there

either. This is an extreme example, but it illustrates how high loads or resource outages could severely impact a configuration that relies only on a follow controller.

Using a scalable controller can help the configuration adjust dynamically in such situations. If the network is online on node1, while the file system is online on node2, then the database can be switched independently as shown in Figure 6.

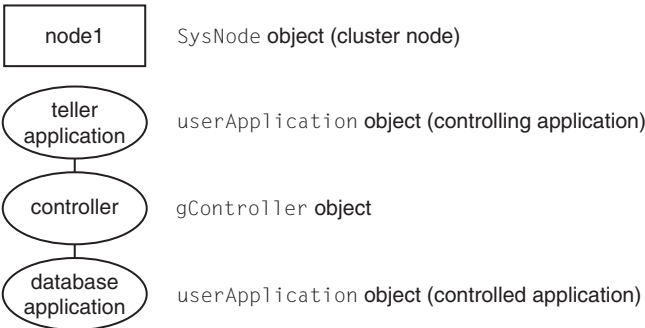


Figure 6: Scalable mode child (controlled) application switchover

Conversely, a network outage could cause RMS to switch the teller to node2 while leaving the database online on node1, as shown in Figure 7.

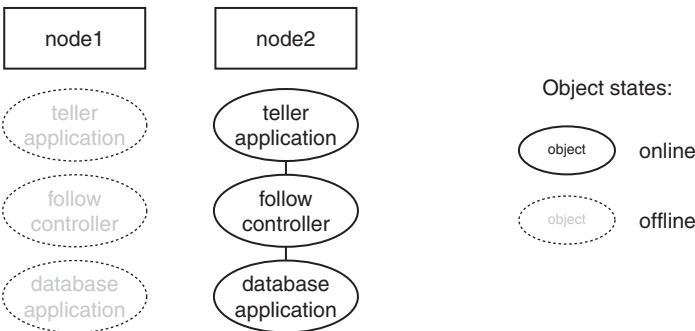


Figure 7: Scalable mode parent (controlling) application switchover

If an application can run on more than one node, RMS allows only one instance of that application to be online in the cluster. That is, an application can run on only one node at a time. However, controller objects do not have the same

restriction. Note the state of the controller objects in Figure 6 and Figure 7. For each scalable child application, an instance of its controller is online on every node where that application can run.

2.2.4.3 Further notes about controllers

The follow and scalable modes are mutually exclusive: a controller for a child application can operate in either follow mode or scalable mode, but not both. The Wizard Tools ensure that each controller's configuration is self-consistent.

A parent application can have more than one child application, but the Wizard Tools require the controller type to be the same for all children. If follow mode is used, then there will be one distinct follow controller for each child. If scalable mode is used, there will be one scalable controller with multiple children. For example, suppose the teller application in the banking scenario also has an automated teller machine (ATM) controlled application. The difference between the follow graph and the scalable graph is shown in Figure 8.

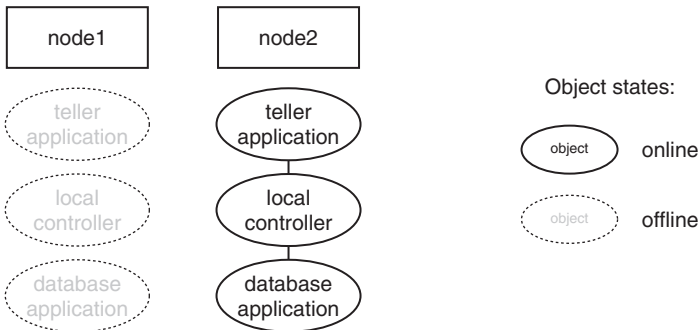


Figure 8: Controller structure with multiple children

For a more detailed discussion, see the section "Scalable controllers".

2.3 How the Wizard Tools provide easy configuration

RMS is a mature product with many features and options. Experts who develop, debug, and fine tune complete RMS configurations must know how RMS works and what RMS needs in order to function properly. For each application in the configuration, the expert must do the following:

- Define the set of resources used by the application, including:
 - Disks
 - Volume managers
 - File systems
 - processes to be monitored
 - IP addresses
- Define the relationship between each resource and its dependent resources, *e.g.*, which file system depends on which virtual or physical disk, which processes depend on which file systems, and so forth.
- Define the relationship between the applications being controlled; for example, which applications must be up and running before others are allowed to start.
- Provide scripts to bring each resource online and offline.
- Provide a detector to determine the state of each resource.

Configuring the above set of requirements by hand can be quite time consuming and prone to errors. This is why the RMS Wizard Tools were developed.

The PRIMECLUSTER RMS Wizard Tools allow the creation of flexible and quality-tested RMS configurations while minimizing your involvement. A simple user interface prompts you for details regarding your applications and resources. Using these details, the Wizard Tools automatically select the proper scripts and detectors and combine them in a pre-defined structure to produce a complete RMS configuration.

Specialists skilled in popular applications and in RMS worked together to create the RMS wizards. The wizards are designed to easily configure RMS for certain popular applications such as Oracle and they are flexible enough to create custom RMS configurations that can control any other type of application.

2.4 RMS wizard products

The RMS wizards are divided into the following separate products:

- RMS Wizard Tools—user interface, general-purpose application wizards, and basic set of subapplication wizards. Provided as a standard component of RMS

- RMS Wizard Kit—set of custom wizards designed to configure specific applications. Available as additional product.

Figure 9 depicts the relationship between RMS, the Wizard Tools, and the RMS Wizard Kit.

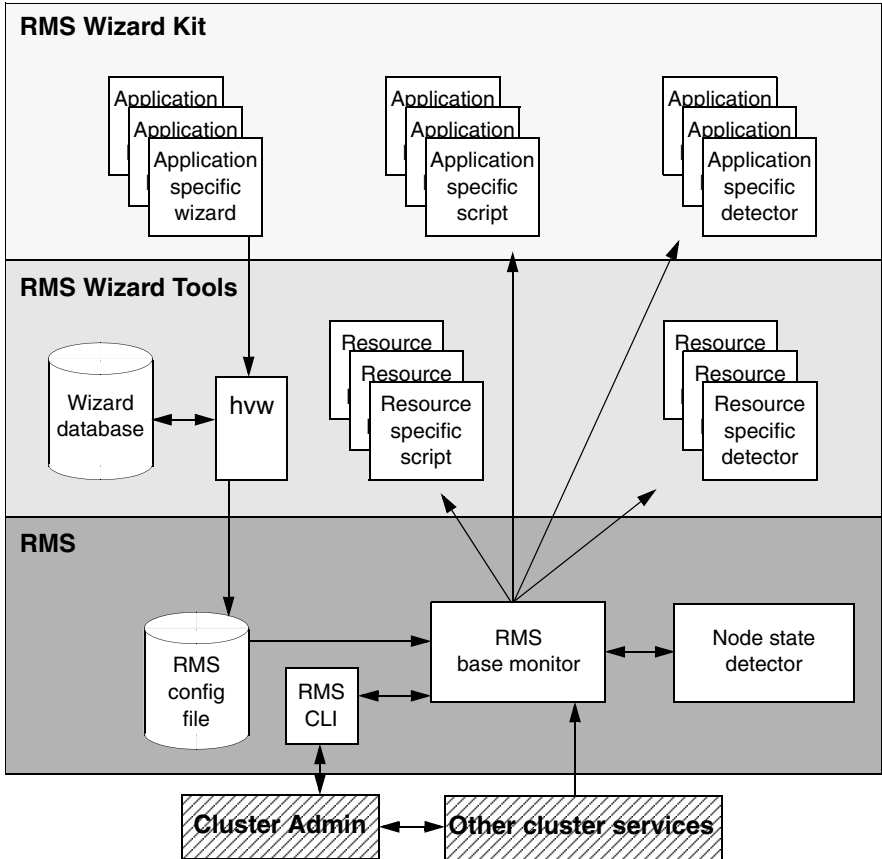


Figure 9: Relationship between RMS and RMS Wizards

For Solaris, as a means of setting the RMS configuration, the userApplication Configuration Wizard, which is a GUI utilizing Web-Based Admin View, is provided.

2.4.1 RMS Wizard Tools

The RMS Wizard Tools provides the following for basic resource types (such as file systems and IP addresses):

- Online scripts
- Offline scripts
- Detectors

In addition to the basic resource support, the RMS Wizard Tools package contains the `hvw` command, which is the entry point to the user configuration interface. The `hvw` interface provides a simple menu-driven interface to allow a user to enter information specific to applications placed under the control of RMS. `hvw` also provides an interface through which application-specific knowledge can be dynamically added to provide turnkey solutions for those applications typically found in the data center. These application-specific modules are provided by the RMS Wizard Kit.

2.4.2 RMS Wizard Kit

The RMS Wizard Kit provides application knowledge modules which can be used by the `hvw` command. The knowledge modules provide `hvw` with information specific to popular applications, which greatly eases the configuration task. The following are also provided for specific applications:

- Online scripts
- Offline scripts
- Detectors



For information on the availability of the RMS Wizard Kit, contact your local customer support service.

2.5 Cluster Admin administration tool

The Cluster Admin GUI is the primary administrative tool for RMS. It allows users full access to the application control functions of RMS, including the following:

- RMS startup and shutdown
- Application startup and shutdown
- Manual application reset and switchover
- Visual cues for resource and application fault isolation
- Fault clearing capability
- Graphs of application and resources
- Logs for nodes and applications
- Maintenance mode control

2.6 RMS components

The RMS product is made up of the following software components that run on each node in the cluster:

- Base monitor
- Detectors
- Scripts
- RMS CLI

2.6.1 Base monitor

The base monitor process is the decision-making segment of the RMS process group. It has the following functions:

- Stores the current configuration of resources as represented by objects, their attributes, and their interdependent relationships
- Receives user requests for specific actions from the Cluster Admin graphical user interface (GUI) or the RMS command line interface (CLI)

- Receives input from detectors and monitors state changes
- Launches scripts to bring applications and their dependent resources online or offline
- Dictates the sequencing of the resource state changes to ensure resources and applications are brought online or offline in the correct order
- Initiates and controls automatic application switchover in case of a resource or node failure, or when directed by a user request
- Performs various administrative functions

2.6.2 Detectors and states

Detectors monitor each object status and return these object statuses to the base monitor.

The detectors that monitor the functions provided by OS are provided by RMS Wizard Tools. Additional application-specific detectors are provided by RMS Wizard Tools and RMS Wizard Kit.

Some objects provided by RMS, such as the controller object, have no detector. Instead, RMS calculates the state of the object based on factors such as transitory processes and the states of its dependent resources.

2.6.3 Scripts

RMS uses scripts to perform actions such as moving a resource from one state to another (for example, from `Offline` to `Online`). The two types of scripts are as follows:

- Request-triggered scripts initiate a state change to a resource.

The request-triggered scripts are as follows:

- `InitScript`— It is executed only once when RMS is started.
- `PreCheckScript`— It is executed to determine if `Online` or `Standby` processing is needed or possible.
- `PreOfflineScript`— It is executed before a transition to the `Offline` state.
- `OfflineScript`— It is executed to set a resource to the `Offline` state.

- `PreOnlineScript`— It is executed before a transition to the `Online` or `Standby` state.
- `OnlineScript`— It is executed to set a resource to the `Online` state.
- State-triggered scripts react to specific events.
 - The state-triggered scripts are as follows:
 - `PostOnlineScript`— It is executed after a transition to the `Online` or `Standby` state.
 - `PostOfflineScript`— It is executed after a transition to the `Offline` state.
 - `OfflineDoneScript`— It is executed after `Offline` processing is completed.
 - `FaultScript`— It is executed when a transition is made to the `Faulted` state.
 - `WarningScript`— It is executed when a transition is made to the `Warning` state.
 - `StateChangeScript`— It is executed when the state of the controlled child application and the state of the node where the child application is running transition.

Scripts that control the functions provided by OS are provided with RMS Wizard Tools.

2.6.4 RMS CLI

The primary interface for configuring RMS is the RMS Wizard Tools, and the primary interface for administering RMS is the Cluster Admin GUI. Both the RMS Wizard Tools and Cluster Admin call the RMS CLI. For example, use the following CLI command to manually switch the `userApplication` object to another node (`SysNode`) in the cluster.

► `# hvswitch userApplication SysNode`

Table 1 lists the RMS CLI commands available to administrators. For a complete description of any command's usage, see its online man page. For a list of all commands related to RMS, see "Appendix B Manual Pages" in "PRIMECLUSTER Installation and Administration Guide."



With few exceptions, RMS CLI commands require root privilege. The exceptions are noted in the following table.

Command	Function
hvassert	Tests an RMS resource for a specified resource state. It can be used in scripts when a resource must achieve a specified state before the script can issue the next command. Does not require root privilege.
hvcem	<p>Starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the hvcem command.</p> <p>The base monitor is the decision-making module of RMS. It controls the configuration and access to all RMS resources. If a resource fails, the base monitor analyzes the failure and initiates the appropriate action according to the specifications for the resource in the configuration file.</p>
hvconfig	<p>Either displays the current RMS configuration or sends the current configuration to an output file.</p> <p>The output of the hvconfig command is equivalent to the running RMS configuration file, but does not include any comments that are in the original file. Also, the order in which the resources are listed in the output might vary from the actual configuration file.</p>
hvdisp	Displays information about the current configuration for RMS resources. Does not require root privilege.
hvdispall	Displays the resource information of all the nodes in RMS.
hvdump	Gets debugging information about RMS on the local node.
hvlogclean	Either saves old log files into a subdirectory whose name is the time RMS was last started, or, if invoked with the -d option, deletes old log files. In either case, hvlogclean creates a clean set of log files even while RMS is running.

Table 1: Available CLI commands

Command	Function
hvsetenv	<p>Provides an interface for changing the following RMS environment variables on the local node:</p> <ul style="list-style-type: none"> – HV_RCSTART controls the automatic startup of RMS. – HV_AUTOSTARTUP controls the automatic startup of all applications. <p>Changes made by this command apply only to the command execution node. For more information about these environment variables, see "Appendix—Environment variables".</p>
hvshut	<p>Shuts down RMS on one node or all the nodes in the configuration. The base monitor on the local node sends a message to other online nodes indicating which node or nodes will be shut down.</p>
hvswitch	<p>Manually switches control of a <code>userApplication</code> or a <code>gResource</code> between <code>SysNodes</code> in the RMS configuration.</p>
hvutil	<p>Provides general administration interface to RMS. This command performs the following management operations:</p> <ul style="list-style-type: none"> – Dynamically setting logging levels – Sending the <code>userApplication Offline</code> – Sending the <code>gResource Offline</code> – Clearing faulted resources – Stopping cluster nodes in the Wait state – Setting detector time periods – Setting Maintenance Mode, and so forth

Table 1: Available CLI commands



Excluding some commands (such as `hvshut`), `hv_*` commands of RMS only send requests to the base monitor then return immediately. They do not wait for the requests to be processed. The status code 0 (success) at the end of command execution means that the requests have been successfully sent to the base monitor. However, this does not secure that the requests have been processed successfully.

2.7 Object types

An object type represents a group of similar resources that are monitored by the same detector (for example, all disk drives). Using the Wizard Tools, you can create configuration files that contain objects of various types, each representing resources or groups of resources to be monitored by RMS. The supported types are as follows:

- SysNode
- userApplication
- gResource
- controller
- andOp
- orOp

Refer to the "Appendix—Object types" for the supported types, their required attributes, and a brief description of each object.



This information is provided for reference only. These objects are created by the Wizard Tools during the generation phase of the configuration process. The type of an object may be listed in diagnostic messages for use by RMS experts.

2.8 Object attributes

An attribute is the part of an object definition that specifies how the base monitor acts and reacts for a particular resource during normal operation. An attribute can include a device name and configuration scripts. Users can specify attributes in any order in the object definition.

Refer to the "Appendix—Attributes" for the supported types, their associated values, and a description of each attribute.



This information is provided for reference only. The values are determined by the Wizard Tools during the generation phase of the configuration process. Refer to the "Using the Wizard Tools interface (hvw)".

2.9 Environment variables

RMS uses global and local environment variables:

- Global environment variables must have the same setting on all the nodes in the cluster. RMS maintains global environment variables in the `ENV` object.
- Local environment variables can vary from node to node. RMS maintains local environment variables in the `ENVL` object.

RMS creates the `ENV` and `ENVL` objects dynamically from the contents of `hvenv` and `hvenv.local` when the base monitor starts up. Refer to the chapter "Setting environment variables" for more information.



- Global variable settings (`ENV`) are included in the configurations checksum that is common to the cluster. The checksum is verified on each node during startup of the base monitor. RMS will fail to start if it detects a checksum difference between the values on any two nodes.
- Local variable settings (`ENVL`) are not included in the configurations checksum that is common to the cluster.

While RMS is running, you can display the environment variables with the `hvdisp` command which does not require root privilege. Use `hvdisp ENV` for global list and `hvdisp ENVL` for local list.

Refer to the "Appendix—Environment variables" for a complete description of each of these variables.

2.9.1 Setting environment variables

When RMS starts, it reads the values of environment variables from `hvenv` and `hvenv.local` and initializes the `ENV` and `ENVL` objects respectively. To change the values of environment variables before RMS starts up, you need to specify them in the `hvenv.local` file.

These settings will be overridden at installation.



The `/tmp` directory that is nearly full may result in RMS errors, because the base monitor uses the `sort(1)` command to sort RMS environment variables.

If you adjust a global variable in the `hvenv.local` file on one node, you must make the same adjustment to `hvenv.local` on every other node in the cluster. After stopping RMS on all the nodes, you will have to restart RMS for the change to take effect.



Caution

Do not explicitly set RMS environment variables in the user environment. Doing so can cause RMS to lose environment variables settings.

The values of environment variables are specified as `export` directives in each file. A typical `export` directive would appear as follows:

```
export SCRIPTS_TIME_OUT=200
```

Therefore, any changes you make to `hvenv.local` file will not take effect until the next time RMS starts up. While RMS is running, you can display the environment variables with the following commands:

- `hvdisp ENV`
- `hvdisp ENVL`

2.9.2 Script execution environment variables

When the RMS invokes a script on behalf of an object, it provides a set of variables in the script's environment that can be used for decision processing at runtime. Since these variables exist only within the context of the script while it is carrying out its tasks, they are not usually visible in the RMS user or administrator environment. In rare cases, they could appear in a diagnostic message in the system log or on the console.

The section "Script execution environment variables" provides a complete description of each of these variables.

2.10 RMS Directory structure

RMS software consists of a number of executables, scripts, files, and commands, all located relative to the directory specified in the `RELIANT_PATH` environment variable. Table 2 illustrates the directory structure of the RMS software after it has been correctly installed.

Name	Contents
<code>RELIANT_PATH</code>	Base directory. Default: <code>/opt/SMAW/SMAWRrms</code>
<code><RELIANT_PATH>/bin</code>	Executables, including detectors, commands, and scripts.
<code><RELIANT_PATH>/build</code>	Work and storage area for configuration files.
<code><RELIANT_PATH>/etc</code>	Miscellaneous files used by RMS and the configuration tools.
<code><RELIANT_PATH>/include</code>	RMS include files (header files) used by detectors and configuration files.
<code><RELIANT_PATH>/lib</code>	RMS runtime libraries.
<code><RELIANT_PATH>/us</code>	RMS source files. The names of the files in this directory are reserved and should not be used to name any configuration files that the user may create.

Table 2: RMS base directory structure

As summarized in Table 3, RMS log files are located in the directory specified in the `RELIANT_LOG_PATH` environment variable.

Name	Contents
RELIANT_LOG_PATH	<p>Contains files that can be used for RMS analyzing and debugging, including the RMS switchlog. The base monitor and detectors create log files here. Default: /var/opt/SMAWRrms/log</p> <p>The same directory has subdirectories that contain backup copies of the RMS log files. Each backup subdirectory has a name of the form <i>yyyy-mm-dd_HH:MM:SS</i> to indicate the date and time when the backup was created.</p>

Table 3: Log directory structure

3 Advanced RMS concepts

This chapter describes details of RMS. It includes the following sections:

- "State transition" deals with the RMS state engine.
- "RMS heartbeat operation" describes the monitoring of node states using heartbeats and processing when a node is down.
- "Scalable controllers" describes operational details of scalable controllers and scalable applications.

3.1 State transition

This chapter deals with the RMS state engine. In particular, it describes how states are determined, and how RMS causes state changes and reacts to state changes.

- "Internal organization"
- "Initializing"
- "Online processing"
- "Offline processing"
- "Fault processing"
- "Switch processing"
- "Special states"

3.1.1 Internal organization

A brief description of the object-oriented internal aspects of the base monitor is useful in understanding RMS.

Every object is an independent instance that carries out actions (typically implemented by shell scripts) according to rules based on its state and messages received from detectors or other objects. States, detectors, and scripts are briefly described in the section "Introduction". The following sections provide more details about RMS internal structure and inter-object communication.

3.1.1.1 Application and resource description

The configuration wizards generate a description for all applications that will be monitored by RMS. The description, which is maintained in an RMS-specific meta-language, represents every application with a logical graph that has the following characteristics:

- Resources required by the applications are represented by objects in this graph.
- Parent/child relationships between objects represent interdependencies between resources.
- Object attributes represent the properties of the resources and the actions that are required for specific resources.

The proactive procedures that bring a particular object online or take it offline are specified by referring to shell scripts that are configured as attributes of the object. Other script attributes specify actions to be taken in reaction to state changes of the object as a result of messages from other objects.

A `userApplication` object has no detector, and if it has been configured by the Wizard Tools or the RMS Wizard Kit, it has no scripts specified. Instead, a child `cmdLine` resource is configured with the appropriate scripts, and it is this object that interacts with the actual user application in the operating system environment. In this case, the `userApplication` becomes a logical container that represents the combined states of the resources in its graph.

3.1.1.2 Messages

RMS objects exchange messages for the following purposes:

- To send requests
- To communicate changes in the object states

In general, objects communicate only with their direct parents and children.

RMS sends incoming external requests to the parent `userApplication` object before it forwards the requests to the children. A `userApplication` object can also generate its own requests on the basis of changes to its state (such as a change over to the `Faulted` state).

3.1.2 Initializing

After RMS starts, the initial state of all objects is `Unknown`. RMS changes this state after the object has the necessary information for identifying the actual state.

The following is necessary information for identifying the state:

- For objects with a detector—First report of the detector
- For objects with children—Messages of the children concerning their state

Two conclusions can be drawn from the above:

- Leaf objects without a detector are illegal in an RMS configuration, because they cannot generate a detector report and they are not able to logically derive their state from the state of their children. Their state always remains `Unknown`.
- All transitions from the `Unknown` state are always bottom-up, such as from the leaf object to the `userApplication`. Every object above the leaf object first requires the state of its children before it is able to determine its own state.



`SysNode` objects and `userApplication` objects have no detectors for their physical counterparts. `SysNode` objects receive a detector report directly from the base monitor. `userApplication` objects determine their status from their children.

After the `userApplication` object exits the `Unknown` state, the initializing process of the application ends. From this point, RMS controls the application.

The initializing processes of `userApplication` objects are independent of each other. Therefore, one `userApplication` object may be initialized to an `Online`, `Offline`, or `Standby` state while a second `userApplication` object is still in the `Unknown` state.

The initializing process of `SysNode` objects is also independent. A `SysNode` object exits its initial `Unknown` state after receiving its detector report.

The `Unknown` state is a pure initial state. Once an object exits the `Unknown` state, it does not return to that state.

Example configuration

The examples of RMS processing in the following sections are based on an application `app` configured to run on `fuji2RMS` and `fuji3RMS` as follows:

- For each node, `fuji2RMS` and `fuji3RMS`, there is one `SysNode` object bearing the name of the node.
- For each `SysNode` where a particular user application may run, the corresponding `userApplication` object has one child of type `andOp`, which bears the name of this `SysNode` as the `HostName` attribute. The order in which the nodes were defined in the `userApplication` object determines the priority of the nodes for this application.

The base monitor identifies each `andOp` object at this level as a **local** object if the value of the `HostName` attribute corresponds to the local node's name, and as a **remote** object if not. The base monitor ignores all remote objects, so that only the local objects and its children are processed.

- As children of this logical *AND* object, the other resources (a command line subapplication and a local file system) are configured according to their internal dependencies.

A diagram of the object hierarchy is shown in Figure 10.

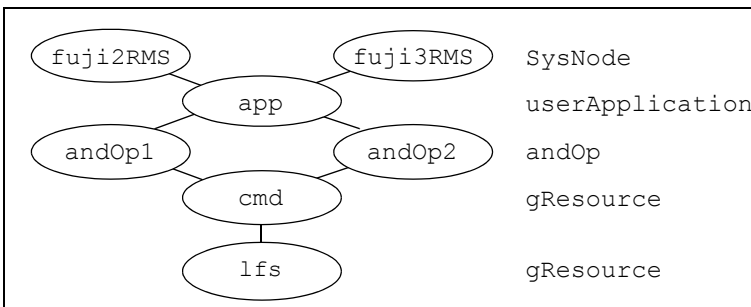


Figure 10: Object hierarchy for initializing examples

Note that the hierarchy in the figure includes the `SysNode` objects as parents of the application. While this is often done by convention, these `SysNode` objects are not dependent on the application objects in any way; however, their presence serves as a reminder of which nodes are represented by the application's child `andOp` objects. They also appear in the graph generated by the GUI, but in this case their major purpose is to indicate the node where the application is currently running.

The actual RMS graph for this configuration as produced by RMS Wizard Tools is shown in Figure 11.

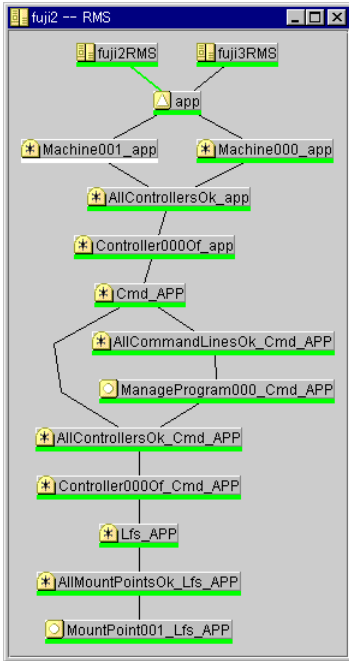


Figure 11: System graph for initializing examples—RMS Wizard Tools

The title bar contains the name of the node on which the graph was drawn, which is fuji2RMS. The line connecting fuji2RMS to app is green, indicating the node where the application is online.

The corresponding output from 'hvdisp -a' is shown in Figure 12.

```
# hvdisp -a
Local System: fuji2RMS
Configuration: /opt/SMAN/SMANRms/build/config.us

Resource          Type      HostName      State      StateDetails
-----
fuji3RMS          SysNode           Online
fuji2RMS          SysNode           Online
app               userApp           Online
Machine001_app   andOp    fuji3RMS
Machine000_app   andOp    fuji2RMS           Online
ManageProgram000_Cmd_APP gRes              Online
MountPoint001_Lfs_APP gRes              Online

#
```

Figure 12: hvdisp output for initializing examples—RMS Wizard Tools

To match the `hvdisp` output, the actual graph includes resource names, which can be displayed by selecting *Preferences* → *Show Resource Names in Graph* in the Cluster Admin *rms* tab view.

Note that some of the actual object names generated by the RMS Wizard Tools are more complex than the simplified, generic names shown in the abstract graph of Figure 10. Also, the graph contains additional dependencies that are automatically inserted to ensure proper operation. Selecting other items from the Cluster Admin *Preferences* menu will display further detail and reveal additional objects and dependencies.

Neither the additional objects nor the complex names in the actual graph are required for a basic understanding of RMS operation. Therefore, to simplify the discussion, the examples in this chapter will focus on the abstract graph and use the generic names.

Example 1

The following process for the configuration illustrated in Figure 10 is applicable for a monitor running on `fuji2RMS`:

1. RMS starts.
2. The base monitor determines the state of the `SysNode` objects.
3. The detectors of the `cmd` and `lfs` resources report their respective states as `Offline`.
4. Since it is a leaf object, `lfs` changes immediately to `Offline` and reports this state change to its parent.

5. After receiving the detector report and the report of its child, `cmd` possesses the necessary information for determining its own state. It then goes offline and notifies this change of state to its parent `andOp1`.



`andOp2` is a remote object which is ignored by the base monitor on `fujii2RMS`.

6. `andOp1` is a logical object which has no detector. It uses the message of the child to determine its own state as `Offline` and notifies this change of state to `app`.
7. `app` is also an object without a detector. When the child `andOp` that corresponds to the local node goes offline, `app` also goes offline.
8. All local child objects of `app` have exited the `Unknown` state and the initializing procedure is complete.

3.1.3 Online processing

The online processing for a `userApplication` object normally results in the `userApplication` transitioning to the `Online` state. Online processing of one `userApplication` object is independent of the online processing of any other `userApplication` objects.

The following situations can prevent successful online processing of a `userApplication`:

- The `PreCheckScript` determines that the `userApplication` should not come online.
- A fault occurs during online processing.

These situations are discussed in detail in later sections.

3.1.3.1 Online request

Generating the online request is referred to as **switching** the `userApplication`; that is, switching the `userApplication` online or switching the `userApplication` to another cluster node (refer also to the "Switch processing").

The following actions can generate an online request:

- Manual request using the GUI or CLI (`hvswitch`)

- Automatic request when RMS is started using the GUI or CLI (hvcm)
- Automatic requests controlled by the application's `AutoSwitchOver` attribute:
 - `AutoSwitchOver` includes `ResourceFailure` and a fault occurs
 - `AutoSwitchOver` includes `ShutDown` and a node is shut down
 - `AutoSwitchOver` includes `HostFailure` and a node is killed

Manual methods

Manual methods have two modes for switching the `userApplication`. These modes are as follows:

- **Priority switch**—RMS selects the `SysNode`. The `userApplication` is switched to the highest priority `SysNode`. The `SysNode` objects' priority is determined by their order in the `PriorityList` attribute of the `userApplication` object.
- **Directed switch**—The user selects the `SysNode`. The `userApplication` is switched to a specific `SysNode`.

In both priority and directed switches, only `SysNode` objects that are in the `Online` state may be selected.

Manual request using the GUI

To manually generate an online request, perform the following steps:

1. Using the graph, right-click on an application to display its context menu.
2. Click on a *switch* or *online* item in the context menu.

Manual request using the CLI

To generate an online request for each `userApplication`, use the `hvswitch` command. Refer to the `hvswitch` manual page for details on usage and options.

Automatic methods

All automatic methods can only invoke a priority switch.

Automatic request at RMS startup

When RMS first starts on a cluster, it switches the `userApplication` online on the highest priority node if all of the following conditions are true:

- All `SysNode` objects associated with a specific application are online.
- The `userApplication` is neither online nor inconsistent on any other cluster node.
- The `AutoStartUp` attribute of the `userApplication` is enabled.
- No object in the graph of the `userApplication` is in the faulted state.

These limitations ensure that the `userApplication` is not started on more than one cluster node at a time.

If the `userApplication` is already online after startup, an automated startup request for the `userApplication` is immediately created, even if `AutoStartUp` is not set or not all `SysNodes` are online. This is intended to ensure a consistent graph of an online `userApplication`. Otherwise objects could still be offline in an graph of an online application.

Automatic request when a fault occurs

RMS initiates a priority switchover when it detects either a fault of a `userApplication`, or a fault of a `SysNode` where a `userApplication` was online. This automatic switchover is controlled by the application's `AutoSwitchOver` attribute as follows:

- `AutoSwitchOver` includes `ResourceFailure` and a fault occurs
- `AutoSwitchOver` includes `ShutDown` and a node is shut down
- `AutoSwitchOver` includes `HostFailure` and a node is killed

No automatic switchover occurs if `AutoSwitchOver` is set to `No`.

3.1.3.2 PreCheckScript

The `PreCheckScript` is intended to verify in advance that certain prerequisites for successful online processing are fulfilled. It avoids useless attempts when those prerequisites are not (yet) met. The `PreCheckScript` is also invoked during policy-based switching.

The `PreCheckScript` will be forked before the original online processing begins. If the script is successful and returns with an exit code of 0, online processing proceeds as usual. If the script fails and returns with an exit code other than 0, online processing is discarded and a warning is written into the switchlog.

Resulting state

When the `PreCheckScript` is running, the `userApplication` object transits into the `Wait` state. If the `PreCheckScript` fails, the `userApplication` object transits back into its previous state, usually `Offline` or `Faulted`.

AutoSwitchOver

If the `PreCheckScript` fails and the `AutoSwitchOver` attribute includes `ResourceFailure`, then RMS automatically forwards the online request to the next priority node (except in cases of directed-switch requests).

3.1.3.3 Online processing in a logical graph of a userApplication

If the `PreCheckScript` is successful, the base monitor generates a **pre-online request**. Relative to the resource graph, the pre-online request process is as follows:

1. Request is sent from the parent to the child.
2. Parent object changes to the `Wait` state, but no script is initiated.
3. Child receives the request. The pre-online script is initiated in the leaf objects.
4. When the script terminates, confirmation is sent to the parent.
5. As soon as all children of the parent have sent their confirmation, the pre-online script is executed on the parent.

In relation to the resource graph, the above steps illustrate the **bottom-up procedure** for executing the scripts in online processing.

The `userApplication` object is the final object to execute its pre-online script; it then generates an online request that is passed to the leaf objects. However, there is a difference between online processing and pre-online processing.

Relative to the resource graph, the online script process is as follows:

1. RMS executes the online script.

2. The system waits until the object detector reports the `Online` state. If an object does not have a detector, the post-online script executes after the `OnlineScript` is completed successfully.
3. The post-online script executes immediately.
4. Confirmation of the success of online processing is forwarded to the parent.
5. The object exits the `Wait` state and changes to the `Online` state.

In the context of RMS, "*the userApplication is online*" means that all configured resources are online (ready to operate). In this case, the term online does not pertain to the state of the actual application. The actual application is started and monitored by the scripts configured for a `cmdline` child object.

i How a `cmdline` script influences the state of the actual application depends on the application itself. RMS has no direct control over any user application. For a more complete discussion, see the section "Relationship of RMS configurations to the real world."

Example 2

The scenario for this example is as follows:

- `AutoStartUp` attribute is set to "yes."
- None of the resource objects have `PreOnlineScript` definitions.
- All objects are in the `Offline` state at startup time.

Online processing is as follows:

1. RMS starts.
2. `userApplication` object `app` on node `fuji2RMS` generates a pre-online request because the `AutoStartUp` attribute is set to "yes."
3. This request is passed through to the `lfs` leaf object. As no `PreOnlineScript` has been configured for any of the objects in this example, `lfs` forwards a message to `app` indicating that pre-online processing has completed successfully.
4. When the pre-online success message arrives, `app` generates the online request, which is also passed through to the `lfs` leaf object.
5. The `lfs` object executes the online script and brings the disk online.

6. As soon as the detector of `lfs` reports `Online`, successful completion of online processing is notified upwards to the `cmd` object. (If the object had a post-online script, this would have been executed before the success message was forwarded.)
7. The `cmd` object starts its online script.
8. As soon as the `cmd` detector reports a success completion, the success message is forwarded to `andOp1`.
9. The `andOp1` object is an object without a detector; it does not have an online script in this example. As soon as its local child reports the `Online` state, it forwards the success message to its parent object `app`.
10. Upon receipt of the success message at `app`, RMS executes the online script and the application starts. Because `app` does not have a detector and also because no post-online script is configured, `app` changes immediately to the `Online` state after the online script has completed successfully.

3.1.3.4 Unexpected reports during online processing

Unexpected reports during `online` processing mean reports which are reported during the `online` processing but not in the `Online` state ignored by the base monitor.

Reports other than in the `Online` state ignored by the base monitor may be reported from the point where the `online` processing of the user application which an object belongs to starts until the object's `online` processing succeeds or fails.

For cases where the object's `online` processing fails, see the "Fault situations during online processing".

3.1.3.5 Fault situations during online processing

If an error situation occurs during `online` processing, the affected object commences `fault` processing and notifies its parent of the error (see also the "Fault processing"). The following can cause faults during `online` processing:

- a. When the last reported state from a detector of a resource object is in the `Offline` or `Faulted` state at the point where `Online` processing finishes.
- b. Script fails with an exit status other than 0.
- c. Script fails with a timeout.

- d. An object's `OnlineScript` finishes and the detector does not notify the `Online` state within a specific period.

For case a, fault processing is initiated after online processing of `userApplication` finishes in direct contrast to cases b, c, and d where fault processing is initiated immediately once that condition is satisfied.

3.1.3.6 Initialization when an application is already online

A situation can occur in which the entire logical graph of a `userApplication` is already online when RMS is initialized. In this case, the `PreCheckScript` does not execute and the affected objects switch directly from the `Unknown` state to the `Online` state without executing any scripts.

Request while online

If a `userApplication` receives an online request when it is already online, it is forwarded to the other objects as usual. The only difference from the description in the "Online processing" is that any objects that are already online forward the request or the responses without executing their scripts and without changing to the `Wait` state. In particular, the `PreCheckScript` is not run.

A typical example of an object which is always online when RMS is initialized is a `gResource` object for a physical disk, since physical disks cannot in general be disabled through a software interface.

No request while online

If a `userApplication` does not receive an online request when it is already online and RMS is initialized, the `userApplication` carries out online processing of its graph as if it had received an explicit online request. The resulting state of the local graph is exactly the same as in the previous case.

Guarding against data loss when the application is already online

A primary objective of RMS is to ensure that no data loss occurs as a result of simultaneous activity of the same application on more than one node in the cluster. Therefore, after the online processing of the application's graph in either of the two cases described above, the base monitor on the local node reports the `userApplication` object's `Online` state to the base monitors on the other nodes to ensure that no corresponding application goes online elsewhere in the cluster.



It can be extremely damaging if a `userApplication` is online on more than one node immediately after RMS has initialized. In this case, RMS generates a `FATAL ERROR` message and blocks any further requests for the `userApplication`. This minimizes the possibility of damage caused by inconsistency in the cluster.



Caution

The situations described in this section are a result of manual intervention. If the manual intervention allowed competing instances of an application or a disk resource to run on multiple nodes, data corruption may have already occurred before RMS was initialized.

3.1.4 Offline processing

Normally, offline processing results in the `userApplication` object transitioning to the `Offline` state.

3.1.4.1 Offline request

An offline request can be generated for any of the following reasons:

- Manual offline request using the GUI or CLI (`hvutil -f`)
- Manual switch request using the GUI or CLI (`hvswitch`)
- Offline processing after a fault, either automatically or using the GUI or CLI (`hvutil -c`)
- RMS shutdown using the GUI or CLI (`hvshut`)

In normal operating mode, only the RMS command interface can generate an offline request. In the case of a fault, the `userApplication` generates its own offline request (such as if one or more necessary resources fails); this prevents an application that is no longer operating correctly from continuing to operate in an uncontrolled manner (see also the "Fault processing"). This offline request is also a primary precondition for any subsequent switchover.



Offline processing of `userApplication` objects does not occur if RMS is shut down with `'hvshut -L'` or `'hvshut -A'`.

3.1.4.2 Offline processing in a logical graph of a userApplication

Unlike online processing, the direction of offline processing is from the userApplication to the leaf object (top-down). Nodes without a detector execute the post-offline script immediately after the offline script. The offline process is as follows:

1. The userApplication changes to the Wait state.
2. The userApplication executes its pre-offline script, and sends a corresponding request to its children after the pre-offline script terminates.
3. After receiving the pre-offline request, each child object changes to the Wait state, executes its pre-offline script, and forwards the request.
4. As soon as the leaf objects have completed their pre-offline script, they send a corresponding message (confirmation of successful pre-offline processing) to their parents.
5. The message is forwarded without any further activity from the children to the parent until it arrives at the userApplication.
6. After pre-offline processing has been completed, the userApplication executes its offline script, immediately followed by the post-offline script (userApplication is an object without a detector).
7. The userApplication then generates the actual offline request.

Processing of the offline request in the individual objects is similar to online processing, as follows:

- The offline script is executed first.
- The post-offline script is started after the object's detector Offline report has arrived.
- After the post-offline script has completed, the offline request is forwarded to each of the object's children.
- When all children have returned a PostOfflineDone message, the object returns a PostOfflineDone message to its parent.

As illustrated, the userApplication is the final object to go offline. After the last child returns a PostOfflineDone message, the offline processing is complete; the OfflineDoneScript, if present, is fired; and the base monitor notifies the corresponding userApplication objects on the other nodes that the application has gone offline.

Example 3

The following further explains the offline process:

1. As none of the objects in the example has a pre-offline script, the corresponding pre-offline request is forwarded from `app` down to the leaf object.
2. The leaf object returns a success message to the `userApplication`.
3. The `userApplication` executes its offline script; in our example, this means that the application `app` is stopped. As the object `app` does not monitor the application, RMS considers the successful completion of the offline script to be a successful completion of offline processing.
4. A post-offline script is not configured, and an offline request is accordingly sent to `andOp1` immediately after the offline script has completed.
5. The `andOp1` object has no detectors and no scripts. The offline request is simply permitted to pass through.
6. The `cmd` object executes its offline script and forwards the request as soon as its own detector signals that offline processing has completed successfully.
7. The `lfs` leaf object also executes its offline script and forwards the success message after the corresponding report of its detector.
8. Offline processing completes successfully when `app` receives the success message.
9. Upon successful completion of offline processing, the `OfflineDoneScript` is fired. This script is intended for cleanup or for sending information. Its return code has no impact on the state of the `userApplication`.

3.1.4.3 Unexpected reports during offline processing

Unexpected reports during `offline` processing mean reports which are reported during the `offline` processing but not in the `Offline` state ignored by the base monitor.

Reports other than in the `Offline` state ignored by the base monitor may be reported from the point where the `offline` processing of the user application which an object belongs to starts until the object's `offline` processing succeeds or fails.

For cases where the object's `offline` processing fails, see the "Fault situations during offline processing".

3.1.4.4 Fault situations during offline processing

If an error situation occurs during `offline` processing, the affected object commences `fault` processing and notifies its parent of the error (see also the "Fault processing"). The following can cause faults during `offline` processing:

- a. When the last reported state from a detector of a resource object is in the `Online`, `Standby`, or `Faulted` state at the point where `Offline` processing finishes.
- b. Script fails with an exit status other than 0.
- c. Script fails with a timeout.
- d. An object's `OfflineScript` finishes and the detector does not notify the `Offline` state within a specific period.

3.1.4.5 Object is already in Offline state

An object may already be `offline` at the start of `offline` processing. This typically occurs if an `offline` request originates from a host where the parent `userApplication` is `offline`. (If the parent `userApplication` is `online`, then the `offline` object must be in the tree below an `OR` object.) When an `offline` object receives an `offline` request, the request is merely passed through, similar to the situation in `online` processing. Scripts are not executed, and the `Wait` state is not entered.

3.1.4.6 Object cannot be sent to Offline state

RMS covers an extremely wide range of system conditions, including monitoring resources that cannot be taken to the `Offline` state by a script. Physical disks are an example of such objects because they are monitored but cannot in general be physically shut down. For this purpose, RMS provides the attribute `LieOffline` to indicate that the resource has no true `Offline` state. The Wizard Tools subapplications set this attribute by default for `gResource` objects that represent physical disks, so it does not have to be explicitly specified.

During `offline` processing, an object whose `LieOffline` attribute is set reacts in the same way as any other object when its `pre-offline`, `offline`, and `post-offline` scripts are run. The reaction of the object with respect to its parent is also the same as if the object had been successfully taken `offline`; that is, it "lies." An object with `LieOffline` set does not wait for an `offline` report of the detector

after the offline script has executed; instead, it automatically executes the post-offline script. An unexpected online report of the detector (which arrives after the offline script has executed) is not a fault condition in this case.

3.1.5 Fault processing

The handling of fault situations is a central aspect of RMS. How RMS reacts to faults differs depending on the state of an application at any particular time. For instance, the reaction to faults that occur in the resource graph of an ongoing application differs from the reaction to faults in the graph of an application that is locally offline.

3.1.5.1 Faults in the online state or request processing

When a detector indicates a fault for an online object whose corresponding `userApplication` is also online, RMS executes the fault script of the object. An equivalent fault condition occurs if the detector indicates that a previously online object is offline although no request is present.

After the fault script completes, RMS notifies the parents of the fault. The parents also execute their fault scripts and forward the fault message.

A special case is represented by `orOp` objects, which report a logical *OR* of their children's states. These react to the fault message only if no child is online. If any child of the parent `orOp` is online, RMS terminates the fault processing at this point.

If there is no intermediate `orOp` object that intercepts the fault message, it reaches the `userApplication`. The `userApplication` then executes its fault script. There are three possible cases during processing according to the following combinations of the `AutoSwitchOver` and `PreserveState` attributes:

- `AutoSwitchOver` includes `ResourceFailure`
- `AutoSwitchOver` does not include `ResourceFailure` and `PreserveState=1`
- `AutoSwitchOver` does not include `ResourceFailure` and `PreserveState=0` or is not set

AutoSwitchOver includes ResourceFailure

When the `AutoSwitchOver` attribute includes `ResourceFailure`, RMS ignores the `PreserveState` attribute and responds as if only the `AutoSwitchOver` attribute were set. In this case, the process is as follows:

1. The `userApplication` attempts to initiate the switchover procedure. For this purpose, the application on the local node must be set to a defined `Offline` state. The procedure is the same as that described under offline processing.
2. When offline processing is successfully completed, an online request is sent to the corresponding `userApplication` of a remote node (see the "Switch processing"). However, the `userApplication` is now in the `Faulted` state—unlike the situation with a normal offline request. This prevents the possibility of an application returning to the node in the event of another switchover.

If a further fault occurs during offline processing; for example, if RMS cannot deconfigure the resource of an object that was notified of a `Faulted` state, then it does not execute a switchover procedure. RMS does not execute a switchover because it views the resources as being in an undefined state. The `userApplication` does not initiate any further actions and blocks all external, non-forced requests.



A failure during offline processing that was initiated by a previous fault is called a double fault.

This situation cannot be resolved by RMS and requires the intervention of the system administrator. The following principle is applicable for RMS in this case: Preventing the possible destruction of data is more important than maintaining the availability of the application.

If the application is important, the `HaltFlag` attribute can be set in the `userApplication` during the configuration procedure. This attribute ensures that the local node is shut down immediately if RMS cannot resolve a double fault state, provided there is another node available for the application. The other nodes detect this as a system failure, and RMS transfers the applications running on the failed node to the available node.

AutoSwitchOver does not include ResourceFailure and PreserveState=1

In this case, the process is as follows:

1. The `userApplication` does not initiate any further activity after the fault script executes.
2. All objects remain in their current state.

Use the `PreserveState` attribute if an application can remedy faults in required resources.

AutoSwitchOver does not include ResourceFailure and PreserveState=0 or is not set

In this case, RMS carries out offline processing as a result of the fault, but it does not initiate a switchover after offline processing is complete (successful or not).

Fault during pending switch request

A special case occurs when a switch request causes a fault during offline processing. In this case, RMS carries out a switchover after completing the offline processing that the fault caused (provided that offline processing is successful), even if the `AutoSwitchOver` attribute is set to `No`. Switchover had evidently been requested at this time by the system administrator who sent the switch request online. If the ongoing switch request is a direct switch request, the target node of the switchover procedure may not be the node with the highest priority; it is the node explicitly specified in the directed switch request.



For more information about the `AutoSwitchOver` and `PreserveState` attributes, see the "Appendix—Attributes".

3.1.5.2 Offline faults

Even if an application is not online on a node, RMS still monitors the objects configured in the application's graph. If a detector indicates a fault in one of these objects, the fault is displayed. However, no processing takes place, the fault script is not executed, and no message is sent to the parent.

In this case, it is possible that an `andOp` object could be offline, even though one of its children is `Faulted`.

This design was chosen on the principle that mandatory dependencies between the objects in a `userApplication` graph exist only if the `userApplication` is to run.

3.1.5.3 AutoRecover attribute

An object of the type `gResource` that represents a local file system is one example of an object that can enter a `Faulted` state due to reasons that are easily and automatically remedied. A fault that occurs in the object itself (and not as a result of an input/output fault on an underlying disk) is most likely from a `umount` command that was erroneously executed. In this case, causing the entire application to be switched over probably would not be the best remedy. Therefore, fault processing would not be the best solution.

For such cases, administrators can configure an object's `AutoRecover` attribute. If a fault then occurs when the object is online, the online script is invoked before the fault script. If the object enters the `Online` state again within a specific period after the online script has been executed, fault processing does not take place.

RMS only evaluates the `AutoRecover` attribute when the object is the cause of the fault, that is, when the cause of the fault is not the fault of a child. Accordingly, RMS only evaluates `AutoRecover` for objects with a detector. The `AutoRecover` attribute is not relevant if a fault occurs during request processing or if the object is in the `Offline` state.

3.1.5.4 Fault during offline processing

A fault occurrence during offline processing does not result in an immediate halt of offline processing at that object. Instead, the fault condition at that point in the tree is stored, and offline processing continues in the normal manner down to the leaf objects. However, the fault is recalled and handled when the success/failure message is propagated to the object on the way upstream to the `userApplication`. This design avoids race conditions that could occur if the fault were processed immediately.

3.1.5.5 Examples of fault processing

The following are examples of fault processing.

Example 4

The scenario for this example is as follows:

- The application `app` has its `AutoSwitchOver` attribute set to `No` and is online on node `fujii2RMS`.
- There is no request.

- The `lfs` object does not have its `AutoRecover` attribute set.
- An error by the system administrator unmounts the `lfs` file system.

Fault processing is as follows:

1. The `lfs` object's `gResource` detector indicates that its object is offline. Because the corresponding `userApplication` is online and because there is no offline request, RMS interprets this offline report as a fault and notifies the parent `cmd`.



Reminder: An unexpected `Offline` state results in a fault.

2. The `cmd` object in this example does not have a fault script. The `cmd` object goes directly to the `Faulted` state and reports the fault to its parent `andOp1`.
3. `andOp1` does not have a fault script either, so it also goes directly to the `Faulted` state, and reports the fault to the parent `app` object.
4. The `app` object then changes to the `Faulted` state and starts offline processing in preparation for switchover, since its `AutoSwitchOver` attribute is set to a value other than `No`.
5. In this example, assume that the local file system `lfs` uses the mount point `/mnt`, and the offline script of `lfs` consists of the simple instruction `umount /mnt`. Because `/mnt` is no longer mounted, this offline script terminates with an exit status other than `0`.
6. Accordingly, offline processing for RMS fails after a fault. A switchover is not possible because the local state remains unclear. RMS waits for the intervention of the system administrator.

A more complex offline script for `lfs` could check whether the object is still mounted and terminate with an exit status of `0`. In this case, RMS could successfully complete offline processing after the fault and switch over to `fuj i3RMS`; all local objects on `fuj i2RMS` would then be offline following successful online processing, and only `app` would remain in the `Faulted` state.

Example 5

The scenario is the same as in the previous example, except the `AutoRecover` attribute is set for the `lfs` object.

Fault processing is as follows:

1. The `lfs` object's `gResource` detector indicates that its object is offline. Since the corresponding `userApplication` is online and because there is no offline request, RMS interprets this offline report as a fault (see *Example 4* above).
2. Since the `AutoRecover` attribute is set, RMS does not immediately report the fault to the parent `cmd` object. Instead, RMS starts the `lfs` object's online script to reverse the unmount procedure.
3. A few seconds later, the `lfs` object's `gResource` detector reports that the object is once again online. RMS returns the object to the `Online` state, and no further fault processing takes place.

Example 6

In this scenario, `app` receives an online request, but the file system represented by `lfs` has been corrupted.

Fault processing is as follows:

1. Online processing starts as a result of the request.
2. The `lfs` object starts its online script, which terminates with an exit status other than 0.
3. The `lfs` object then initiates fault processing: it starts its fault script (if one is configured), changes to the `Faulted` state, and notifies RMS of the fault.
4. The rest of the process proceeds in the same manner as described in *Example 4* above.



Fault processing in this case would be the same even if the `AutoRecover` attribute were set. This attribute is only significant if the application is in a stable `Online` state, that is, the application is online and there is no pending request.

3.1.5.6 Fault clearing

After successful offline processing due to a fault occurrence, the resource objects will be offline, and the `userApplication` object will be faulted. If offline processing fails as a result of the fault, or if the application's `PreserveState` attribute is set, at least part of the graph may remain in a state other than `Offline`, *i.e.*, `Online`, `Standby`, or `Faulted`.

In all of the above states, the `userApplication` prevents switch requests to this host, because the base monitor assumes that at least some of the resources are not available. After the system administrator has remedied the cause of the fault, one of the following procedures can be used to notify the base monitor so that RMS can resume normal operation:

1. The following command may be used to clear the faulted state of the `userApplication` object and the objects in its graph:

```
hvutil -c userApplication
```

This command attempts to clear the fault by switching the parent application and its graph into a self-consistent state: if the application object is online, then online processing will be initiated; if the application object is offline, then offline processing will be initiated. (The user is notified about which type of processing will occur and given a chance to abandon the operation.) The fault clears successfully when every branch leading to the application reaches the same online or offline state. If the final state is offline, the system administrator can set the `userApplication` to the online state with a switch request.

If the `userApplication` object is initially online, invoking `'hvutil -c'` may not affect every object in the tree. To initiate offline processing for the entire tree, use `'hvutil -f'` as described below.

2. The following command initiates an offline request to the `userApplication` object:

```
hvutil -f userApplication
```

This starts offline processing for the application. If the command completes successfully, the application and every object in its graph are switched to the offline state, and the fault is cleared. If required, the system administrator can set the `userApplication` to the online state with a switch request.

In summary, if the `userApplication` is in the faulted state, both `'hvutil -f'` and `'hvutil -c'` have the same effect: both result in **offline** processing.

The difference occurs when the `userApplication` is online and a fault occurs below an `orOp: 'hvutil -f'` would initiate **offline** processing for the `userApplication`; but `'hvutil -c'` would act as if `'hvswitch -f'` had been invoked, and **online** processing for the `userApplication` would begin.

3.1.5.7 SysNode faults

RMS handles a fault that occurs in a `SysNode` in a different manner than faults in any other type of object. A `SysNode` fault occurs under the following conditions:

- The local base monitor loses the heartbeat of the base monitor on a remote host.
- A CF `LEFTCLUSTER` event occurs

When either of these events happen, RMS must first ensure that the remote node is actually down before automatic switchover occurs. To accomplish this, RMS uses the Shutdown Facility (SF). For more information about the Shutdown Facility and shutdown agents, see "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide."

Once the shutdown of the cluster node is verified by the SF, all `userApplication` objects that were `Online` on the affected cluster node, and whose `AutoSwitchOver` attribute includes `HostFailure`, are priority switched to surviving cluster nodes.

Example 7

The scenario for this example is as follows:

- RMS is running on a cluster consisting of nodes `fuji2` and `fuji3`, which are represented by the `SysNode` objects `fuji2RMS` and `fuji3RMS`, respectively.
- `app` is online on `fuji2RMS` and its `AutoSwitchOver` attribute includes the `HostFailure` setting.
- A system fault on `fuji2` generates a panic message.

The reaction of RMS is as follows:

1. CF determines that a node failure has occurred and generates a `LEFTCLUSTER` event.
2. RMS puts the `SysNode` in a `Wait` state. RMS receives the `LEFTCLUSTER` event and sends a kill request to SF.

3. After SF successfully kills the node, a `DOWN` event is sent.
4. RMS receives the `DOWN` event and marks the `SysNode` as `Faulted`.
5. The `fuji2RMS` object executes its fault script (assuming that such a script has been configured).
6. The `fuji2RMS` object notifies the `userApplication` objects that `fuji2RMS` has failed. Since `app` was online on `fuji2RMS` when `fuji2RMS` failed, and since its `AutoSwitchOver` attribute includes the `HostFailure` setting, the object `app` on `fuji2RMS` starts online processing.

Operator intervention

If the Shutdown Facility is engaged to kill a node, but the duration of the `SysNode` object's `Wait` state exceeds the object's `ScriptTimeout` limit, RMS records an `ERROR` message in the switchlog to this effect.

At this point, one cluster node is now in an undefined state, so RMS blocks all further action on all other nodes. This situation is usually resolved only by operator intervention as described in "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide." Upon successful completion of the procedure, CF sends a `DOWN` event, RMS resolves the blocked state, and normal operation resumes.

For more information about the `ScriptTimeout` attribute, see the "Appendix—Attributes".

3.1.6 Switch processing

The switch processing procedure ensures that an application switches over to another node in the cluster.

3.1.6.1 Switch request

Switch requests are divided as follows:

- Priority switch request—RMS identifies the target node according to the node priority list defined during the configuration process.
- Directed switch request—The user specifies the target node.

The types of switches are divided as follows:

- **Switchover**—The application running on a node is to be switched over to another node.
- **Switch-online**—An application that is not running on any node is started; or the node on which it has previously been running has failed.



During switch processing, RMS notifies all the nodes in the cluster of the procedure. This prevents competing requests.

Example 8

The scenario for this example is as follows:

- app is online on fuji2RMS.
- The system administrator sends a directed switch request on fuji3RMS with the aim of switching app to fuji3RMS.

Switch processing is as follows:

1. RMS forwards the switch request to fuji2RMS because fuji2RMS is the online node of the userApplication object.
2. app on fuji2RMS notifies the corresponding nodes in the cluster (in this case, app on fuji3RMS) that switchover processing is active. This means that competing activities are blocked.
3. app on fuji2RMS sends a request to app on fuji3RMS to establish whether any faults are known in the local graph on fuji3RMS. In this example, there are no known faults in the local graph on fuji3RMS.
4. app on fuji2RMS commences offline processing.
5. As soon as RMS has successfully deconfigured app on fuji2RMS, app on fuji2RMS notifies all corresponding nodes in the cluster that the application will now be running on fuji3RMS. Blocking of competing requests is simultaneously cancelled.
6. app on fuji2RMS terminates its activity by sending an explicit online request to app on fuji3RMS.
7. app on fuji3RMS commences online processing.

3.1.7 Special states

3.1.7.1 Restrictions during maintenance mode

An application in maintenance mode imposes processing restrictions on other objects that appear in the same graph. These restrictions prevent all processing described in earlier sections of this chapter, and they affect the following objects:

- All child objects of the application in maintenance mode
- All ancestors up to and including the node where the application in maintenance mode resides

The restrictions may generate either an error or a timeout, depending on the type of processing request.

To illustrate how the restrictions apply, consider the situation in which two applications (app1 and app2) each have a single dependent resource (Res1 and Res2, respectively) and are configured to run on either of two nodes (NodeA and NodeB). Figure 13 shows the states of these objects if app1 is put into maintenance mode.

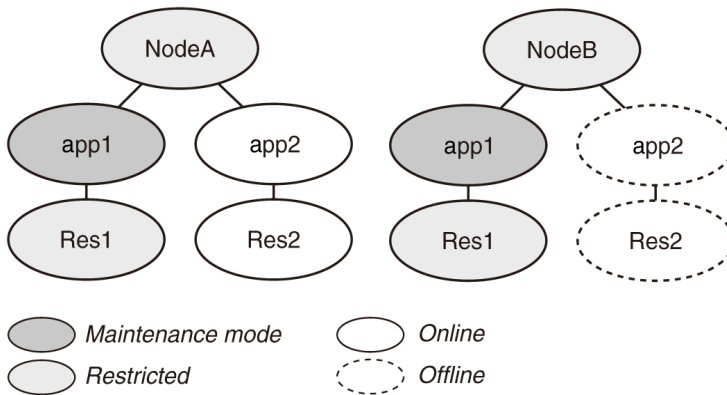


Figure 13: Example of maintenance mode restrictions

This simple example illustrates the following features:

- Maintenance mode for an application applies on every node where the application could be brought online, *i.e.*, on every node in its *PriorityList* attribute. Therefore, app1 is in the Maintenance state on every node where it could run, regardless of its previous state on any of those nodes.

Note that app1 also has an intended status on each of its nodes. This is the status just before the start of maintenance mode. The intended status is available in the application's *StateDetails* parameter in the GUI and in the CLI `hvdisp` output.

- Since Res1 is a child of app1, it is restricted everywhere.
- Since app1 appears in the graph of both NodeA and NodeB, they are also restricted. (You can initiate RMS shutdown of either node, but you will be prompted to let RMS take app1 out of maintenance mode first.)

- `app1` does not appear in the graph of `app2`. Therefore, normal processing of `app2` and its resource is allowed, including switching `app2` offline, online, or to a different node.
- Do not stop RMS or the system while the application is in the maintenance mode. You must exit the maintenance mode of all applications before stopping them.

If you stop RMS by using a `force` option before exiting the maintenance mode of all applications, the OS will shut down without the offline process being performed.

Therefore, this could sometimes result in some resources like volume manager being in an incorrect state when RMS comes back up.

This behavior is typical if an application is put into maintenance mode with `'hvutil -m on'`, or if the equivalent GUI operation begins by right-clicking on the individual application. However, if `'hvutil -M on'` is used, or if the GUI operation begins by right-clicking on the cluster name, then maintenance mode is clusterwide and processing is suspended everywhere.

3.1.7.2 The Inconsistent state

There are situations in which a `userApplication` and one or more of the resources in its graph are `Offline` or `Faulted`, while other resources in its graph are `Online` or `Faulted`. This could be the result of manual intervention by an administrator, or it could occur when a fault clearing operation fails and leaves some objects in `Online` or `Faulted` states.

It may be that some of these `Online` or `Faulted` resource objects have their `ClusterExclusive` attribute set, which indicates that they should not be brought `Online` on two or more hosts at the same time. If the `userApplication` were marked simply as `Offline` or `Faulted`, it could be switched `Online` on another node along with all its resources. This would be in direct conflict with the intention of the `ClusterExclusive` attribute, and the resulting resource conflict might cause data corruption. To avoid problems in these cases, RMS prevents any switch of the `userApplication` by marking it as `Inconsistent` rather than `Offline` or `Faulted`. The exact definition is as follows:

- A `userApplication` is marked with the `Inconsistent` state if its actual state is either `Offline`, `Standby`, or `Faulted`, and one or more resource objects in its graph are either `Online` or `Faulted` and have their `ClusterExclusive` attribute set to 1.

Note that while the `userApplication` is displayed or reported as `Inconsistent`, the actual state is either `Offline`, `Standby`, or `Faulted`. For most operations, the behavior of an `Inconsistent` `userApplication` is determined by the underlying actual state. For instance, if the actual state is `Offline`, and an `Offline` request is issued, no `Offline` script will be fired (see the "Object is already in Offline state").

The exception to this behavior occurs when there is a request to switch an `Inconsistent` `userApplication` to a remote node: in this case, the request is denied. This avoids possible damage by ensuring that the `ClusterExclusive` resources are `Online` only on one host at a time.

If a `userApplication` is `Inconsistent` on only one node, then it is possible to switch it `Online` on that node. However, if it is `Inconsistent` on two or more nodes, then it cannot be switched at all; in this case, the inconsistency must be resolved first, *e.g.*, by bringing all resources into an `Offline` state via `'hvutil -f'` or `'hvutil -c'`.



Caution

If a `userApplication` is `Inconsistent` on multiple nodes, one of its `ClusterExclusive` resources may be `Online` on multiple nodes as well. If this is the case, take appropriate action to shut down the resource gracefully on each node **before** you issue an `'hvutil'` command for the `userApplication`. Depending on the resource type, you may also need to determine if there has been any data corruption.

3.2 RMS heartbeat operation

RMS transmits a UDP **heartbeat** signal at regular intervals. If the elapsed time since the last heartbeat from a node exceeds an adjustable connection timeout, RMS assumes the node has lost connectivity (see "HV_CONNECT_TIMEOUT"). RMS then begins a recovery period for the node. If the node heartbeat is detected during the recovery period, RMS assumes the node is functional and returns it to normal status. However, if RMS receives no heartbeats from the node before the recovery period expires, it assumes the node is down, even if other communication with the node is possible.

Once RMS marks a node as down, it takes a series of steps to ensure application and cluster integrity. First, it is necessary to ensure that the node is truly shut down. Otherwise, the node and its applications could unexpectedly recover later, causing conflicts and data corruption. To avoid these problems, RMS

directs the Shutdown Facility (described later) to eliminate the node. This is often done by rebooting the node or turning off its power, but the exact action depends on which shutdown agents have been configured for the node. Only after the node has been eliminated is it safe for RMS to restart the node's applications elsewhere in the cluster. The process of automatically switching applications from a failed node to a healthy node is called application failover.

Application switchover impacts cluster performance, so it is important to choose a recovery timeout that avoids false detection of node outages. The optimum UDP recovery time depends on the conditions in the cluster. A short recovery period is the best choice to deal with failures of nodes or base monitors. However, a long recovery period allows time for overloaded nodes to respond, which avoids unnecessary shutdowns. If the UDP method is used by itself, these opposing requirements make it difficult to tune the recovery time in large or busy clusters.

Note that the UDP method can be unreliable, because it has three potential points of failure: first, an outgoing request for a response may not get through to the remote node, so it has no reason to respond; second, the remote node may be so busy that it cannot respond within the recovery period, especially if the recovery timeout is set to a low value; third, a response packet may be sent from the remote node, but it may not get through to the local node. In all three cases, the local node cannot take action until the recovery period expires.

To improve cluster response, RMS uses its Enhanced Lock Manager (ELM) as the primary method to determine machine states and connectivity. ELM is not a polling method. Instead, it relies on locks that are managed at the kernel level by the Cluster Foundation. When a node joins the cluster, it creates a lock and holds it as long as its base monitor is running. The lock is released when the node or its base monitor goes down. The state of the locks is available locally on each node, because the Cluster Foundation maintains them in the background.

ELM is designed to address the high priority issue of node or base monitor failures. The UDP heartbeat can therefore be optimized to detect slow node response, with the recovery time set to a relatively large value. This provides an important complement to ELM. A node with an overloaded CPU or network interface may respond so slowly that the underlying Cluster Foundation cannot determine the state of the node's lock. If this condition persists, the UDP heartbeat recovery period eventually expires, and RMS proceeds to shut down the node. ELM's efficiency and reliability make this a very infrequent occurrence.

Experts can manually disable ELM for rolling upgrade or debugging operations (see "HV_USE_ELM"). In this case, when RMS starts up, the expert must also manually adjust the UDP heartbeat recovery timeout to a smaller value with 'hvcn -h <timeout>'. This is necessary to efficiently detect remote base monitor and node outages in the absence of ELM.

3.2.1 Operational details

The ELM mechanism is best illustrated with a few simple examples. The following discussion assumes the cluster consists of two or more nodes named A, B, C, etc.

3.2.1.1 ELM lock management

The lock name for each node has the format "RMS<nnnnnn>", where <nnnnnn> is the 6-digit CF node ID. Therefore, the correspondence of every node's name, CF node ID, and lock name is easily determined. The locks are maintained clusterwide by CF, and any node can request access to any lock.

When a node first requests access to a lock, the request is granted immediately, because the node will initially receive it in the NULL state. This is a special, neutral state that does not conflict with that lock's state on any other node. The local node can then issue a request to ELM to convert the lock to another state. Once the request is granted, the node can hold it in that state, or it can convert it to another state. Those are the only possible operations.

Besides the NULL state, ELM supports only two other states:

- concurrent read (CR)—Multiple nodes can hold the lock in this state. This will prevent other nodes from converting the lock to the exclusive state.
- exclusive (EX)—Only one node in the cluster can hold the lock in this state. This will prevent other nodes from converting the lock to the concurrent read or exclusive states.

The conversion operation is the central control mechanism for ELM. Requesting a conversion that would conflict with other nodes does not cause an error condition. Instead, the request is put into a queue until ELM can grant the request. The requesting process is put in a wait state, and when it reawakens, it knows its request has been granted.

When a node converts a lock back to the NULL state, it effectively releases the lock, and allows other nodes to convert the lock to their desired state. The requests will be granted in the order they were queued.

For example, suppose A has converted its lock to the EX state. B can request access to A's lock and receive it immediately in the NULL state. If B then issues a request to convert it to the CR state, the request will wait until A (or ELM itself) converts the lock to the NULL or CR state, because neither of these conflict with B's request. Therefore, when B successfully converts A's lock, it knows that A is no longer holding the lock in the EX state.

Since a request to access another node's lock is granted immediately and does not affect the ELM lock mechanism, that step is omitted in the following discussions.

3.2.1.2 First node startup

Assume that A is the first node to start RMS. The following sequence occurs:

1. A converts its own lock to EX mode. This happens immediately, since no other nodes have completed their startup at this point.
2. A initiates its UDP heartbeat and waits for responses from other nodes. It does not yet attempt to convert the locks of any other node at this point.

3.2.1.3 Second node startup

Assume that B is the second node to start RMS. The following sequence occurs:

1. B converts its own lock to EX mode. This happens immediately, since A has not requested access to B's lock at this point.
2. B initiates its UDP heartbeat and waits for responses from other nodes. It does not attempt to convert the locks of any other nodes at this point.
3. A detects the first heartbeat from B. A issues a request to convert B's lock to the CR state. Since B holds its lock in the EX state, A's request goes to sleep while it waits in the ELM queue.
4. Since A's request is sleeping, B must be holding its own lock. Therefore, B must be online, and A marks it accordingly. A continues to mark B as online as long as its request remains asleep.
5. B detects the heartbeat from A and executes a similar sequence. B tries to convert A's lock to CR mode. But A holds its lock in the EX state, so B's request goes to sleep while it waits in the ELM queue.
6. As long as B's request remains asleep, B continues to mark A as online.

At this point, both nodes hold their own locks in the EX state, and each has issued a request to convert the other's lock to the CR state. Both requests are sleeping, but that has no effect on anything else either node is doing. However, the unfulfilled request on each node indicates the other base monitor is online.

3.2.1.4 Third and subsequent nodes

Assume C is the third node to start RMS. The sequence of operations with each of A and B are similar to the second node startup above:

1. C converts its own lock in the EX state, initiates its heartbeat, and then waits for the heartbeats from the other nodes.
2. When it first receives the heartbeat from one of the other nodes, it tries to convert that node's lock to the CR state, and the request goes to sleep.
3. As long as the request remains asleep, C marks the other node as online.

C performs steps 2 and 3 whenever it receives a heartbeat from a node for the first time.

At the same time, the other online nodes receive C's heartbeat for the first time, and they execute steps 2 and 3 with C's lock.

When the entire cluster is online, the states of the locks on each node are as follows:

- The node hold its own lock in the EX state.
- The node has issued requests to convert every other node's lock to the CR state, and all of these requests are sleeping.

3.2.1.5 Node or RMS down scenario

When CF has issued a LEFTCLUSTER event for a remote node, or when the base monitor on a remote node goes down, ELM converts that node's lock to the NULL state on every node in the cluster.

For example, suppose node B has just gone down. The sequence of events on node A is typical of every other online node:

1. A's pending request to convert B's lock to the CR state wakes up. A now holds that lock in the CR state.
2. A immediately converts the lock to the NULL state and marks B as being offline.

3. A continues to mark B as offline until it receives a heartbeat from B. At that point, A restarts the lock cycle by requesting to convert B's lock to the CR state.

The same sequence would have occurred if node B's base monitor went down first. The major difference would be that, if the node goes down, the LEFTCLUSTER event would precede the ELM lock release; if the base monitor went down, the ELM lock release would precede the LEFTCLUSTER event. Either condition would cause RMS to initiate a node elimination.

The ELM method proactively alerts the other base monitors when an outage occurs somewhere in the cluster. They do not have to wait for heartbeat timeouts to expire.

Note that ELM handles a graceful shutdown in much the same way, but in this case, the node itself releases the lock. Also, at the RMS level, no node elimination is necessary.

3.2.1.6 Slow response scenario

When a remote node is busy, its base monitor may respond very slowly. ELM is a state-based method and cannot detect this condition. Therefore, RMS depends on the time-based UDP heartbeat to decide when the remote response has become unacceptably slow.

For example, suppose node B is not down, but its base monitor is responding very slowly. The following sequence will occur on one of the nodes in the cluster, which we will be supposed to node A for this discussion:

1. A's request to convert B's lock to the CR state continues to sleep, because B is still up.
2. B's heartbeat period expires (default: 30 seconds), and then its heartbeat recovery period expires (default: 600 seconds).
3. Based on the heartbeat loss, A directs SF to eliminate B. (Note that ELM still detects no problem.)
4. When B is eliminated, ELM releases its lock on every node.
5. A's request to convert B's lock is granted. A immediately releases the lock and marks B as offline.
6. A waits for B's heartbeat, and the lock cycle starts again.

Other nodes may detect the loss of B's heartbeat before their lock request wakes up, in which case they will also initiate a node elimination of B.

This illustrates why ELM needs the UDP heartbeat as a backup. Without UDP, the ELM lock requests could remain in the queue well beyond the point where the remote node provides no useful services.

3.2.2 Default initialization in hvenv

The `HV_USE_ELM` environment variable in `hvenv` is automatically initialized to enable or disable ELM according to the status of CF:

- If CF is not installed, `HV_USE_ELM` is set to 0 and the default UDP heartbeat recovery timeout is 45 seconds.
- If CF is installed, `HV_USE_ELM` is set to 1 and the default UDP heartbeat recovery timeout is 600 seconds.

You can override the default recovery timeout with `'hvcml -h'`.



If the heartbeat recovery timeout is too short, premature node kills may occur even though ELM is operational.

3.2.3 Manually controlling ELM in hvenv.local

ELM may be manually disabled, but this should be done only by experts or consultants. Reasons for disabling ELM include:

- rolling upgrades
- testing or debugging

When you disable ELM, you must change the `HV_USE_ELM` variable and then start RMS with a suitable timeout:

1. In `hvenv.local`, set `HV_USE_ELM=0`

See `"HV_USE_ELM"`.

2. Start RMS with `'hvcml -h <timeout> ...'`

The recommended timeout when ELM is disabled is 45 seconds. See the `hvcml` man page. There is no way to specify the heartbeat timeout when you start RMS from the Cluster Admin GUI.



You can also set `HV_USE_ELM=1` in `hvenv.local`. However, if this setting is in effect when CF is not installed, you will not be able to start RMS. RMS indicates this with a message in the switchlog.

3.3 Scalable controllers

This chapter describes operational details of scalable controllers and scalable applications. It includes the following sections:

- "Controller overview"
- "Scalable controllers and applications"
- "Online/offline processing and state transitions"

3.3.1 Controller overview

Controllers enable an application to monitor and control other applications. The application that includes the controller sub-application is referred to as the **controlling** or **parent** application; the application(s) specified in the controller description are referred to as the **controlled** or **child** application(s). The child application(s) act like resources of the parent application. The controlling application can be thought of as the master application and the controlled applications can be thought of as slave applications.

3.3.2 Scalable controllers and applications

The scalable controller provides a single point of administration, control, and display of information for multiple applications. Applications controlled by a scalable controller are called scalable applications.

For example, by switching online a controlling application with a scalable controller, scalable applications come online in the order prescribed by the controller's `ApplicationSequence` attribute; by switching the controlling application offline its scalable applications come offline in the opposite order. At the same time, the state of the scalable controller reflects a combined state of the scalable applications—this state can be readily viewed via GUI or `hvdisp`.

Scalable controllers and scalable applications are useful for configurations that require complex control and administration for multiple configuration

resources. Some such configurations have already been implemented in the past using multiple follow-type controllers. However, the scalable controller permits the construction of very complex high availability configurations that are treated as a single entity.

3.3.2.1 Scalable applications

A scalable application is any application controlled by a scalable controller. Different scalable applications can be online at the same time on the same or on different hosts. For example, parallel databases like Oracle RAC, Fujitsu Symfoware and IBM DB2 PE (Parallel Edition) are all database servers that may have separate instances running at the same time on different nodes in the cluster. A separate application is configured for each database instance. A top-level application monitors and controls all applications (or database instances). The top-level application is referred to as the “controlling” application. The “database instance” applications are referred to as “controlled” applications. By organizing such components into scalable applications controlled by a scalable controller, one can use this controller as a single point of access to the whole server across the cluster. Of course, these are the most commonly encountered scalable applications in commercial data processing. However, any application that has similar requirements can be configured similarly.

3.3.2.2 Benefits of scalable controllers

A primary benefit of the scalable controller is that it makes it allows configurations in which child applications can (or must) run on different nodes than the parent. It also simplifies administrative operations for multiple applications such as parallel databases.

For example, one method used to configure Oracle RAC is to provide a separate user application on each node. Each user application is managed independently on each node. This configuration works fine for Oracle RAC because it is a shared access database so that when a node on any instance of the cluster is available, the whole database is accessible.

However, in a non-shared database such as IBM DB2 PE, all of the instances of the database must be on-line before the entire database can be accessed. Building a configuration to support DB2 PE can be complicated using traditional follow-mode controllers because of the multiplicity of applications and relationships. However, with the scalable controller these configurations are much simpler to construct.

Various configuration options are available to the user to support multiple applications or parent-child application dependencies that span multiple hosts. The following sections describe the scalable controller in more detail.

3.3.2.3 Attributes for scalable controllers

The Scalable controller attribute controls support for scalable child applications: if the controller's Scalable attribute is set to 1, then all applications listed in its Resource attribute are considered to be scalable applications.

Only a single scalable controller can control a given child application—the child cannot have multiple parent controllers. A given application may contain one or more scalable controllers, and no limitations are imposed on the structure of the application that contains one or more scalable controllers. However, RMS Wizard Tools may restrict the choices available during configuration according to the template application wizard used.

The following figure shows the Cluster Admin view of a typical scalable controller's attributes.

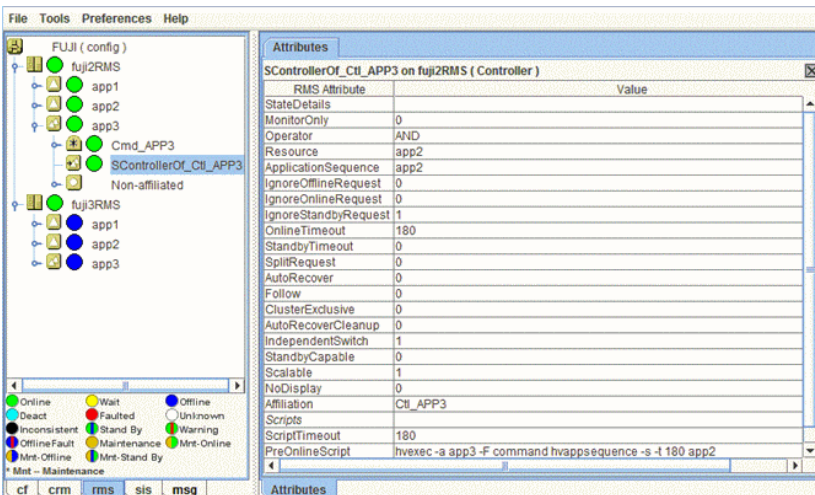


Figure 14: Scalable controller attributes

Some of the scalable controller attributes shown in the figure above can be adjusted in the RMS Wizard Tools interface. These attributes in this example are set as follows (reading top-to-bottom in the figure):

- ✦ MonitorOnly=0
- ✦ Resource=<a list of applications>
- ✦ ApplicationSequence=<the sequence of applications in the Resource list>
- ✦ ScriptTimeout=180
- ✦ FaultScript=<script definition>
- ✦ StateChangeScript=<script definition>

Other attributes that affect scalable operation are set automatically by the RMS Wizard Tools when the user configures a scalable controller.

The effect of some of these attributes on switchover processing and state transitions is discussed in the following sections.

3.3.3 Online/offline processing and state transitions

3.3.3.1 How controller states depend on controlled application states

Table 4 summarizes scalable controller states according to the states of its controlled applications:

Controller state	Child application states
Online	At least one online on any machine
Offline	All offline on all machines
Faulted	At least one faulted on any machine, all others offline on all machines
Standby	At least one standby on any machine, none online on any machine
Warning	At least one online on any machine, others offline on all machines

Table 4: Dependence of scalable controller states on child applications

Note: Although two or more different scalable applications can be online at the same time on the same or on different hosts, each scalable application can still be online only on a single host. For example, when creating an RMS configuration for Oracle RAC, the Oracle daemon processes that must be online on multiple hosts should be represented not by a single RMS application, but by a

number of separate, controlled RMS applications—one child application per daemon. The parent scalable controller for Oracle RAC becomes a single point of control, administration, and display for all these child applications.

3.3.3.2 About nodes of which application become the online state

Controlling applications and controlled applications may become the online state on different nodes. However, monitoring and controlling for applications can work normally and no action is needed.



Note

When you restart a node after it stopped abnormally because of a node failure, an application which is controlled from multiple nodes may become the online state.

However, it has no problem operating the system and no action is needed.

3.3.3.3 Request propagation to controlled applications

Priority `Online` requests from the controller are propagated to all controlled applications, each of which will attempt to come `Online` according to the sequence of hosts in their `PriorityList` attribute. `Offline` and `Standby` requests are propagated to all controlled applications on all the hosts.

The attribute `IndependentSwitch`, which must be set to 1 for a scalable controller object, assures that a parent application can be switched from host to host without affecting controlled child applications. However, the `IndependentSwitch` setting is ignored when the parent application is switched via a forced request such as `hvs switch -f`; in this case the controller will propagate `Offline` request to each child application as a part of its switchover processing.

When a parent application and child applications are switched over simultaneously by shutting down RMS or the operating system and restarting RMS is completed before the child applications status changed to `Online` on the secondary prior node, the child applications statuses will attempt to come `Online` according to the sequence of nodes in their `PriorityList` attributes. Therefore the child applications statuses will be `Online` again on the previous node that has not switched over.

3.3.3.4 Controller warning state

Scalable controller displays `Warning` state in GUI and `hvdisp` command if one of the following conditions is met.

This is to warn administrators that some scalable applications are not running.

- When at least one application, which is controlled by an `Online` controller, was transited to an `Offline`, `Standby`, or `Faulted` state from an `Online` state
- When at least one application, which is controlled by an `Standby` controller, was transited to an `Offline` or `Faulted` state from a `Standby` state



Note

When RMS has been stopped on all the nodes where some applications (among all applications controlled by an `Online` controller) can run, the controller will remain in the `Online` state because it cannot detect a failure.

3.3.3.5 Application warning state

An application that contains one or more scalable controllers will acquire a posted `Warning` state if at least one of its scalable controllers appears in the posted `Warning` state. It will revert to its respective posted state when all its scalable controllers go out of the posted `Warning` state. As was the case for the controller, the GUI and `hvdisp` will display this application posted `Warning` state to alert the administrator that the scalable parent application is not running in its full capacity.

The posted `Warning` state will be changed to a respective posted state when the controller transitions to offline of faulted, or when all controlled applications transition either to online or standby.

3.3.3.6 Controller state change script

The controller's `StateChangeScript` attribute specifies a script to be executed for a scalable controller whenever a controlled child application transitions into an online, offline, faulted, or standby state. The script is executed on every host where the controlling parent application can run, even if it is done on behalf of a request originated from the controller itself. The state change script will also

be executed if a host where a child application is running changes its state to offline or faulted. This script has no impact on state transitions—its abnormal exit code is merely recorded in the `switchlog`.

If more than one state transition event is delivered at the same time, then a state change script will be executed for each one of them. The state change script is scheduled immediately upon receiving an event, without any delay. For multiple events arriving from the same host, it is guaranteed that the order of state change script invocation will follow the order of the received events.

Script execution environment variables

The state change script's environment contains a set of variables that can be used for decision processing at runtime. Their names and purposes are summarized below. For a complete description of their content and format, see the "Global environment variables", the "Script execution environment variables".

RMS provides two complementary variables that are set according to whether the state change script was invoked due to an application state change or a host state change:

- **HV_APPLICATION_STATE_CHANGE_INFO**

Set when a state change script is invoked because a controlled application changed its state. The colon-delimited substrings are:

- Previous state of the application on its node, as recorded on the local node
- Current state of the application on its node, as recorded on the local node
- Name of the node where the application has changed its state
- Name of the application that has changed its state

`HV_APPLICATION_STATE_CHANGE_INFO` will be empty if the state change script is invoked because a node changed its state.

- **HV_HOST_STATE_CHANGE_INFO**

Set when a state change script is invoked for a scalable controller because a `SysNode` that can run a controlled child application changes its state to `Offline` or `Faulted`. The colon-delimited substrings are:

- Previous node state
- Current node state

- Name of the node
- Reason for the node state change: Shutdown Facility, `hvshut` command, or unknown.

`HV_HOST_STATE_CHANGE_INFO` will be empty if the state change script is invoked because a controlled child application changed its state.

Like other scripts invoked by RMS on behalf of an object, the state change script environment includes the following standard set of script variables:

- **HV_APPLICATION**

Name of `userApplication` object at the top of the current sub-tree that contains the current object.

- **HV_AUTORECOVER**

If set to 1, the script was initiated due to an `AutoRecover` attempt.

- **HV_FORCED_REQUEST**

If set to 1, the script is currently processing a forced request.

- **HV_LAST_DET_REPORT**

Last detector report for the current object.

- **HV_OFFLINE_REASON**

Reason for ongoing offline processing: `deact` request, manual switchover, follow-up processing after a previous resource failure, or stopped application.

- **HV_NODENAME**

Name of current object.

- **HV_SCRIPT_TYPE**

Script type.

- **NODE_SCRIPTS_TIME_OUT**

Timeout value for the current object and script type.

Script execution sequence

The state change script is executed immediately upon an event. As a result, it can run in parallel with other RMS scripts, including state change scripts for other objects or even for the current object. Since several events may be

delivered nearly at the same time, the user's script is responsible to work in proper sequence. The writer of the script must assure proper access to shared resources from this and other scripts by using locks or other OS means.

3.3.3.7 Sequenced online/standby/offline and application groups

The controller's `ApplicationSequence` attribute controls how online, offline, and standby requests are propagated to child applications. The attribute lists all the controller's child applications that are specified in the controller `Resource` attribute, but arranged in groups that are processed in parallel or sequentially:

A space-separated group of applications is called a request group. All the applications in a request group are processed in parallel.

Request groups separated by colons (:) are processed sequentially. Online or standby requests are processed left-to-right. Offline requests are processed right-to-left.

For example, suppose the `ApplicationSequence` attribute for a controller contains the following specification:

```
A1 A2:B:C1 C2
```

The controller would issue online requests as follow:

1. A1 and A2 (the first request group) would be processed in parallel—neither would wait for the other. However, both A1 and A2 must post their respective online state before the controller proceeds to the next group.
2. B (the second request group) would be processed next. It must post its online state before the controller proceeds to the next group.
3. C1 and C2 (the third request group) would finally be processed in parallel.

Offline requests would be processed in the reverse order; that is, C1 and C2 first, then B, and finally A1 and A2.

The order is only important during a request propagated to controlled applications from the scalable controller. Applications state changes due to any other reason, including manual or automatic switchover, disregard `ApplicationSequence`.

The order of offline processing is also maintained for local `hvshut` requests such as `'hvshut -l'`. However, the order is disregarded for clusterwide requests such as `'hvshut -a'`: each host will take its applications offline independently.

Like other subapplications, scalable request groups are not processed during 'hvshut -L' and 'hvshut -A' operations.

Note that the propagation of requests from the scalable controller is also affected by the state of the hosts where the child applications may run. If no hosts are available for the applications from the same request group, then the request is not propagated. Instead, it is delayed until such a host becomes available, or until `ScriptTimeout` expires.

3.3.3.8 Auto Startup on a sub-cluster

A controlling application must be able to auto-startup even when some cluster hosts are offline. This is required to allow for some controlled child applications to come online on the partial cluster, in the order defined in their parent controller's `ApplicationSequence` attribute.

For example, if initially all cluster hosts are down, and some of them come up while others are in maintenance, controlled applications that are supposed to run only on the up sub-cluster must be brought online following a request from their scalable controller, regardless of the fact that other cluster hosts are currently offline or faulted. It is impossible to use 'hvswitch -f' in this case because a consultant may not be present, so the configuration must come up on its own.

The above is achieved by the attribute `PartialCluster` of the `userApplication` object. If set to 1, then the application can negotiate its online request within the currently online hosts, even if some other hosts, including the application's primary host, are offline or faulted. If set to 0, then application can negotiate its online request only when all hosts where it can possibly run are online (current behavior). The default value is 0.

For an application that contains a scalable controller (i.e. for a controlling application), `PartialCluster` can be set to 1 if the application graph has no cluster-exclusive resources; otherwise, `PartialCluster` must be set to 0. Each of its controlled applications must have its `PartialCluster` attribute set to 0, and its `AutoStartUp` attribute set to 0.

If controlling applications do not contain any objects other than the controller objects, having their `PartialCluster` attribute set to 1 is safe. Indeed, if two or more controlling applications auto-startup on different sub-clusters, they will not share any exclusive resources. As a part of their auto-startup, they will propagate online requests to their controlled applications that, in turn, may come online if they only require the hosts from the current sub-cluster.

However, if some controlled applications require hosts from several sub-clusters, then these applications will not come online, which is actually the desired effect. Note that their `PartialCluster` is 0, so the controlled applications that span subclusters refuse online processing.

Because scalable controllers delay propagation of the online request until hosts for the controlled applications from each application group come online, the order imposed by `ApplicationSequence` will be preserved even if two or more hosts simultaneously initiate online requests for the same scalable controller.

3.3.3.9 Switchover on a sub-cluster

When a controlling application has its `PartialCluster` set, it can be switched online, automatically or manually (with or without the `-f` forced option) between the hosts of a sub-cluster. Again, this is safe for the controlling application itself because it has no real resources other than controllers. Also, this is safe for the controlled applications because their `PartialCluster` is set to 0—they will not come online if they cross the subcluster boundary.

3.3.3.10 Manual switching of child applications

Attempting to manually take the last online child application offline with the `'hvutil -f'` command would cause the parent controller to go into the faulted state. To prevent the user from creating a faulted scenario, RMS rejects taking the last controlled online application offline with the `'hvutil -f'` command and generates the following message:

```
ERROR: Application controlled from another online application
- <application>.
```

3.3.3.11 Operation during maintenance mode

To illustrate controller behavior in maintenance mode, consider the following example:

- The configuration consists of applications `app1`, `app2`, and `app3` on nodes `shasta1` and `shasta2`.
- `app2` controls `app1` with a follow mode controller.
- `app3` controls `app2` with a scalable mode controller.

Figure 15 shows the RMS configuration tree and the corresponding graph for this scenario.

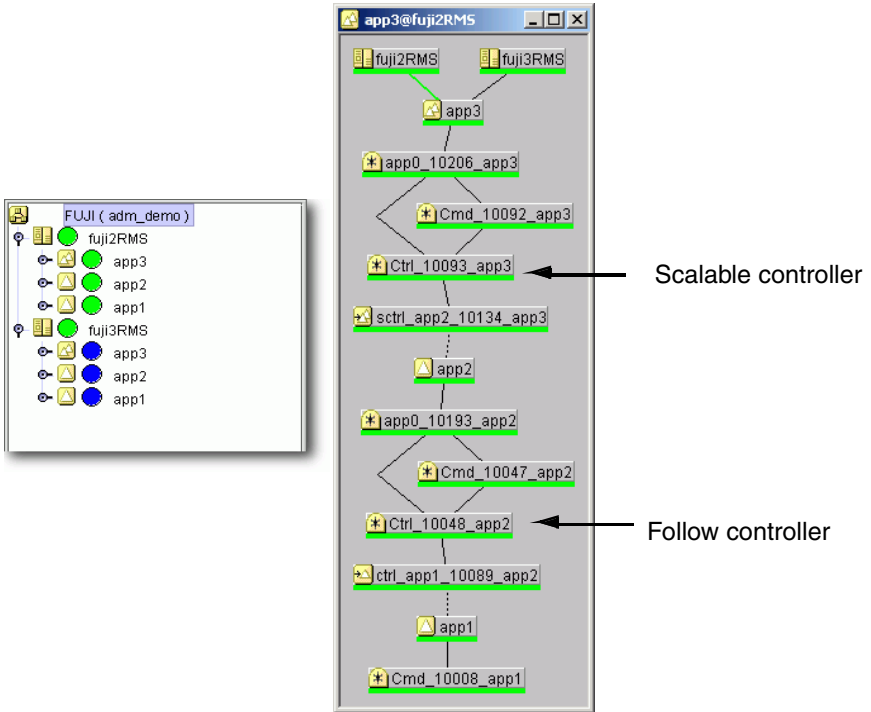


Figure 15: Maintenance mode controller example

Switching applications between nodes

From the perspective of the graph, these may be thought of as “horizontal” switching operations.

If a scalable-mode child application is put into maintenance mode, the parent can still be switched between nodes with `hvswitch`, either manually or automatically (Figure 16). Note that putting `app2` into maintenance mode automatically puts its follow-mode child application, `app1`, into maintenance mode as well.

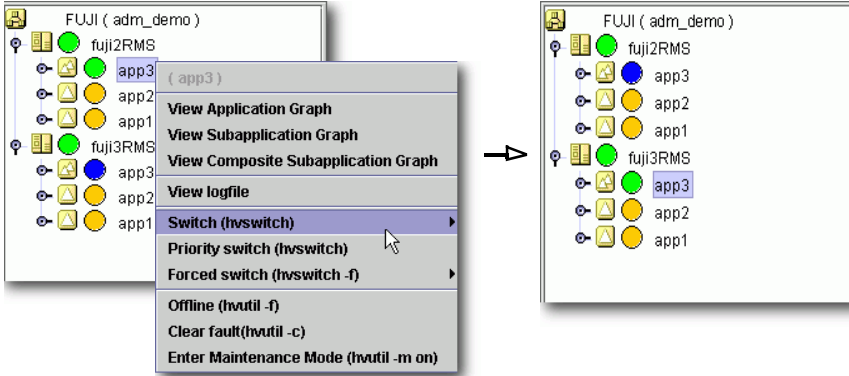


Figure 16: Switching scalable parent with child in maintenance mode

Likewise, if a scalable-mode parent application is put into maintenance mode, the child can still be switched between nodes with `hvswitch` (Figure 17). However, in actual practice, this would only occur via manual commands

because the parent issues no processing requests while in maintenance mode. Note that switching `app2` to a different node automatically switches its follow-mode child application, `app1`, as well.

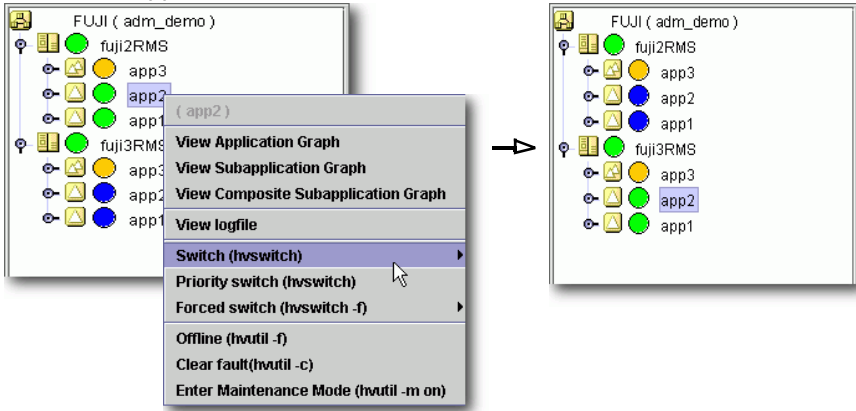


Figure 17: Switching scalable child with parent in maintenance mode

In summary, these “horizontal” hvswitch operations proceed normally.

Online, offline, and fault processing restrictions

From the perspective of the graph, these may be thought of as “vertical” switching operations.

Consider what happens if you attempt to take `app3` offline with `hvutil` while `app2` is in maintenance mode (Figure 18).

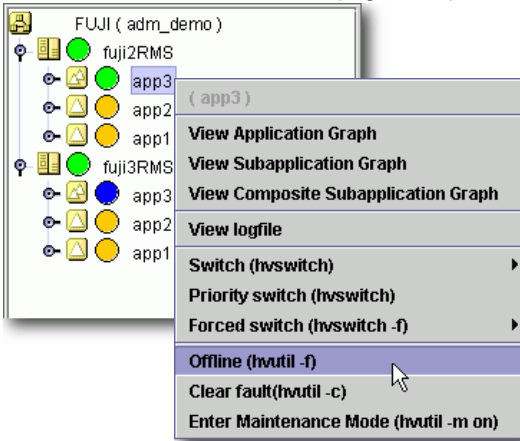


Figure 18: Attempting to take a parent offline with the child in maintenance mode

Unlike the `hvswitch` operations described previously, this `'hvutil -f'` operation requires a response from its child application before it can proceed. Since maintenance mode prevents `app2` from responding to normal processing requests from its parent, the offline request will eventually time out. The same situation occurs when fault processing via `'hvutil -c'` is attempted.

Attempting to take a child offline while the parent is in maintenance mode (Figure 19) also fails, but there is no long delay as in the previous case: RMS immediately generates an error because only the parent can initiate offline processing.

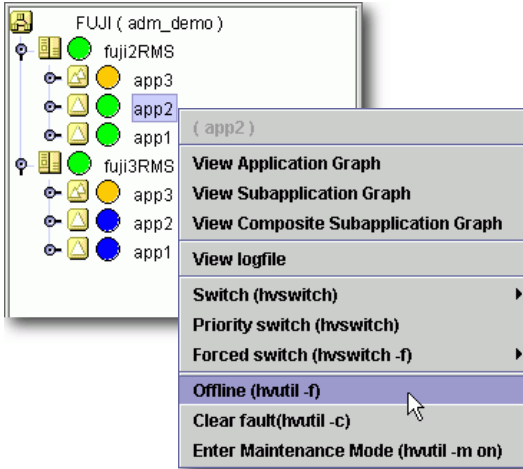


Figure 19: Attempting to take a child offline with the parent in maintenance mode

However, you can issue a fault processing ‘`hvutil -c`’ request to the child because that does not require a response from the parent.

In summary, these “vertical” `hvutil` operations will fail if the application requires a response from a child or parent application that is in maintenance mode.

Recommended approach for scalable controllers

It is highly recommended that maintenance mode requests be issued to the top-level application of a controlled hierarchy. This will help to avoid unexpected behavior:

- Child applications will receive no automatic requests from the parent to switch to other nodes. Also, no automatic online, offline, or fault processing requests will propagate from the parent.
- In case of a resource failure, a switch request from lower-level objects will fail immediately, so RMS can initiate fault processing without waiting for a timeout delay.

4 Using the Wizard Tools interface (hvw)

This chapter describes how to configure high availability for customer applications using the RMS Wizards.

- The "Overview" gives a brief overall description of the configuration process and the RMS Wizards.
- The "General configuration procedure" outlines the four major steps involved in every configuration procedure.
- The "Creating and editing a configuration" describes the wizard interface and how it is used to specify a configuration.
- The "Activating a configuration" describes how to activate a configuration after it has been created or modified.
- The "Configuration elements" provides additional details about basic RMS elements specified in every configuration.



Note

For Solaris, as a means of setting the RMS configuration, the userApplication Configuration Wizard, which is a GUI utilizing Web-Based Admin View, is provided. For this reason, use hvw only when instructed.

For details on the userApplication Configuration Wizard, see "4.4.4 userApplication Configuration Wizard Functions" in "PRIMECLUSTER Installation and Administration Guide (Oracle Solaris)."

All the following procedures assume the Cluster Foundation (CF) software has been properly installed, configured, and started. See "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide" for details.

4.1 Overview

The "Introduction" describes the components necessary for configuring applications for high availability. It is extremely important that you define applications and the resources that are used by them. Resources are entities like disks, file systems, processes, IP addresses, and so forth.

This definition also needs to include the following information:

- How the applications and their resources are related to each other
- What scripts bring resources online and offline
- Which detectors monitor the state of which resources

For example, if a node should fail to be available, the node that is to take its place must have been defined beforehand so that the applications depending on this node are able to continue operating with minimal interruption. Once the necessary information is defined, you can then set up an RMS configuration. A configuration of this magnitude, however, requires a great deal of expert knowledge.

The RMS Wizards are tools that allow you to set up an RMS configuration in a way that is simple, flexible, and quality-tested. Furthermore, these tools conform to a well-documented, standard design. To configure RMS with the wizards, you supply information about the applications using a menu-driven interface. The wizards use this information to set up a complete RMS configuration.

The following sections describe these wizards and the way they are used to configure high availability from a general point of view.

4.1.1 RMS Wizard types

The RMS Wizards are divided into two categories:

- **RMS Wizard Tools**—This is a general-purpose package that includes the following components:
 - The `hvw` menu-based configuration interface
 - **SCALABLE** application wizard, which allows configuration of scalable application.
 - **STANDBY** application wizard, which allows configuration of standby application.
 - The basic set of resource-oriented wizards, which provide scripts and detectors for basic resources such as file systems, volume managers, and IP addresses. They are used by the **SCALABLE** and **STANDBY** wizards as well as components in the Wizard Kit.
- **RMS Wizard Kit**—These application-oriented wizards are designed to cover complete applications and perform their tasks on the basis of the turnkey concept. The **ORACLE** wizard is this type of wizard.



For information on the availability of the RMS Wizard Kit, refer to the manual of each wizard product such as Wizard for Oracle or Wizard for Networker.

4.1.1.1 Turnkey wizards

Turnkey wizards provide predefined structures of resources to monitor almost every basic operating system object. This relieves the user of the tedious task of linking system resources according to their dependencies.

Many turnkey wizards, SCALABLE turnkey wizard, STANDBY turnkey wizard, ORACLE wizard, and so on are designed to configure a specific type of application. All characters of the turnkey wizards are in uppercase.

4.1.1.2 Resource wizards

Resource wizards (sometimes called sub-application wizards) configure lower-level resources such as file systems or IP addresses. They are invoked by turnkey wizards and are not designed to interact directly with the user. Resource wizards have names that begin with one uppercase letter followed by one or more lowercase letters. The following are some of the more important resource wizards:

- **Cmdline**—Configures any generic resource type by specifying `StartScript` (to bring the resource online), `StopScript` (to send the resource offline) and `CheckScript` (to check the state of a resource).
- **Controller**—Configures applications that control other applications.
- **Fsystem**—Configures local or remote file systems.
- **Gds**—Configures disk classes administrated by GDS.
- **Gls**—Configures the resource class of GLS by using RMS environment.
- **Ippaddress**—Configures the IP addresses that are needed for communication over a LAN interface.

4.2 General configuration procedure

RMS configuration always involves these four steps:

► **Stop RMS.**

Refer to the "Stopping RMS". You can use the Cluster Admin GUI or the command line interface from any node in the cluster.

► **Create or edit the configuration.**

Refer to "Creating and editing a configuration" .

► **Activate the configuration.**

Activation includes generation and distribution. See the "Activating a configuration".

► **Start RMS.**

Refer to the "Starting RMS". You can use the Cluster Admin GUI or the command line interface from any node in the cluster.



To avoid network access problems, perform RMS configuration tasks as `root`, and ensure that CF is installed, properly configured, and running as described in "Software Release Guide" and "PRIMECLUSTER Installation Guide" for your operating system.

4.3 Creating and editing a configuration

You can bring up an existing Wizard Tools configuration that is currently activated on the host systems of a cluster. In this case, you might call up the configuration because it is to be modified using the wizards while RMS is stopped. On the other hand, you might want to use the wizards to set up a new configuration. The commands for starting the wizards are as follows:

● **hvw**

Runs RMS Wizard Tools using the last activated configuration stored in the `<RELIANT_PATH>/etc/CONFIG.rms` startup file. If this file does not exist or activation is being done for the first time, RMS creates the default configuration, `config`.

● **hvw -n *configname***

Edits an existing configuration or creates a new configuration using the specified name. The configuration will be stored in the `<RELIANT_PATH>/build/configname.us` startup file.

The sample configuration used for demonstration purposes in this chapter shows how to set up a new configuration called `myconfig` using the STANDBY turnkey wizard. This example would be called up as follows:

```
hvw -n myconfig
```

The `hvw` command is documented in the online manual pages. Refer to the "Appendix—Release information" for additional information.

4.3.1 Using the wizard menus

The `hvw` command produces character-driven menus that guide you in a way designed to be self-explanatory. The following are some of the most frequently used menu operations and items:

- **Selecting items**—This is normally done by typing the number of the item followed by the `[Enter]` or `[Return]` key. Within the menu, a prompting line indicates the kind of input that is required. A `>>` prompt indicates that a string of text should be entered.
- **Responding to messages**—Within the menus, several kinds of messages are displayed. One type of message might be to inform the user about the activities that the wizard has performed; for example, a consistency check that ended in a positive result. Other messages may prompt the user to continue the configuration procedure with a certain activity; for example, choosing an application name.
- **HELP**—This item provides user assistance and is available at the top of every wizard menu.
- **QUIT**—This quits the wizard menu system.
- **RETURN**—This moves one level upward in the menu system; that is, from a subordinate menu to the menu it was called from.
- **SAVE+EXIT** and **NOSAVE+EXIT**—These save or discard your input and then exit. **SAVE+EXIT** will be disabled in read-only mode, and it may be disabled if the configuration is inconsistent at that point.
- **NEXT** and **PREVIOUS**—If all items cannot be listed in a single screen due to many items to be selected, some are displayed in the next screen. Press **NEXT** to go on to the next screen, and press **PREVIOUS** to go back to the previous screen.

4.3.2 Main configuration menu

The *Main configuration menu* appears immediately after a configuration has been called up. This top-level menu shows the state of the RMS cluster by indicating either one the following:

- RMS is inactive
- The list of nodes where RMS is up and running

The *Main configuration menu* changes dynamically at run time depending on whether RMS is running in the cluster and whether the configuration being edited is the current configuration.

If RMS is running anywhere in the cluster, actions that could modify a running configuration are not available. Additionally, the menu items that are available are modified such that no changes can be made to the running configuration.

When RMS is running but the configuration being edited is not the same as the currently active one, the main menu is not restricted except that the *Configuration-Activate* menu option is not available.

4.3.2.1 Main configuration menu when RMS is not active

If RMS is not running anywhere, then the entire top level menu is presented without restrictions. Figure 20 shows the *Main configuration menu* window when RMS is inactive.

```
fujj2: Main configuration menu, current configuration: myconfig
No RMS active in the cluster
1) HELP
2) QUIT
3) Application-Create
4) Application-Edit
5) Application-Remove
6) Application-Clone
7) Configuration-Generate
8) Configuration-Activate
9) Configuration-Copy
10) Configuration-Remove
11) Configuration-Freeze
12) Configuration-Thaw
13) Configuration-Edit-Global-Settings
14) Configuration-Consistency-Report
15) Configuration-ScriptExecution
16) RMS-CreateMachine
17) RMS-RemoveMachine
Choose an action:
```

Figure 20: Main configuration menu when RMS is not active

Menu items

The *Main configuration menu* can perform the following activities when RMS is not running anywhere in the cluster:

- *Application-Create*—Specifies which application to configure for high availability. In addition, this operation specifies all the relevant settings for the application so that it can run in a high-availability configuration monitored by RMS. Among the most important of these settings is the name of the application and the list of nodes on which the application may run.

The user application should be configured to run on multiple nodes for a high-availability configuration.

The wizard assists you by supplying menus with basic and non-basic attributes, assigns values to the attributes, and prompts you if an attribute is mandatory.

By choosing the appropriate turnkey wizard for an application, the wizard will then provide predefined elements, like scripts and detectors, for the application in question. These elements have been developed especially for the respective type of application.

The wizard will also carry out consistency checks at certain stages of the configuration procedure in order to prevent inconsistent applications from running in a high-availability configuration.

- *Application-Edit*—Modifies an existing application.

An existing application can be modified using this menu item. The following modes are available for editing an application:

- Turnkey mode (highly recommended)—Turnkey mode is the default mode. This mode is highly recommended because it simplifies complicated tasks like creating linkages between application and sub-applications.
- Non-turnkey mode—Do not use this mode for creating and editing a normal RMS configuration. Use it only when you are requested from field engineers.

- *Application-Remove*—Removes an existing application from the high-availability configuration.

- *Application-Clone*—Clones an application. This feature is provided for users who want to create a new application that differs only slightly from an existing one. To do this, clone an application and modify only the parts that are necessary to create a new one.



Note

Application-Clone is not available in this version.

- *Configuration-Generate*—Performs the following:
 - Runs consistency checks on the configuration
 - Creates the RMS graph of the configuration and stores it in the *configname.us* file. The graph is a hierarchical description of objects that represent the nodes, applications, and resources used in the configuration.

During the *Configuration-Generate* phase, the wizard indicates the progress with a series of dots on the screen. Each dot represents an application or resource that has been successfully generated.

Normally, you would use *Configuration-Activate* (described below) to generate and distribute the configuration in one step. *Configuration-Generate* provides a way to generate and check a configuration without distributing it to the other nodes in the cluster. This may be useful for testing or debugging (see also the description for *Configuration-ScriptExecution* later in this list).



Configuration-Generate is always available, whether RMS is running or not.

- *Configuration-Activate*—Generates and distributes a configuration.

Selecting this item performs both the generation and distribution phases in one step. The generation phase is described above.

The distribution phase prepares the cluster for RMS, ensuring that all the required data is put into place. The wizard copies the configuration data to every reachable node specified in the configuration and installs all necessary files.



- *Configuration-Activate* is not available if RMS is already running on one or more nodes.
- Do not execute the *Configuration-Activate* menu simultaneously on multiple nodes which constitute the cluster.

- *Configuration-Copy*—Produces a copy of an existing configuration. This is often used to make a backup before an existing, tested configuration is enhanced.
- *Configuration-Remove*—Removes (deletes) any existing configuration.
- *Configuration-Freeze*—Prevents further changes to a configuration. This marks the configuration as read-only so it can be viewed, but not modified.



Configuration-Freeze is password protected: you will be prompted to create a password before the configuration is locked.



Note

Configuration-Freeze is not available in this version.

- *Configuration-Thaw*—Releases the configuration from the frozen (read-only) state so it can be modified.



Configuration-Thaw is password protected: you must enter the correct password before the configuration is unlocked.



Note

Configuration-Thaw is not available in this version.

- *Configuration-Edit-Global-Settings*—Modifies settings that affect the entire configuration. This includes settings for the detectors and the operation mode of the hvw command. This item is also used to specify the alternate interconnects (Alternatelps) for the cluster.
- *Configuration-Consistency-Report*—Provides a consistency check that verifies whether an application is running within a high-availability configuration and has actually been created using the configuration data provided by the respective wizard.



Note

Configuration-Consistency-Report is not available in this version.

The wizard compares the currently activated wizard checksum against the wizard database checksum. One checksum is called the *Live-Info*, the other is called the *BuildInfo*. If both checksums match for an application, it is certified that its running version conforms to what was configured by the wizard.

- *Configuration-ScriptExecution*—Allows administrators to run any script independent of RMS.

By selecting the resources configured for the application, the user can execute the scripts that are to bring the resources online or offline. To see the online scripts being executed, you can go through the resource list, which is displayed for this purpose, in ascending order. The return code indicates the proper functioning of the respective script.



Note

Configuration-ScriptExecution is not available in this version.

- *RMS-CreateMachine*—Defines the list of machines which constitute the cluster. During the activation phase, the RMS configuration will be distributed to all the nodes in this list.

Applications managed by RMS must each be configured to run on one or more machines in this pool. Therefore, complete this step before creating any application.

- *RMS-RemoveMachine*—Removes machines from the list of cluster nodes.

4.3.2.2 Main configuration menu when RMS is running

Wizard Tools menus change dynamically according to whether or not RMS is running in the following locations:

- anywhere in the cluster
- on the local node

If RMS is running on any of the cluster machines, any operation which could potentially modify the currently active configuration is not allowed.

In particular, when RMS is running on the local node, the *Main configuration menu* changes as shown in Figure 21.

```
fujii2: Main configuration menu, current configuration: myconfig
RMS up on: fujii2RMS fujii3RMS
1) HELP                      7) Configuration-Freeze
2) QUIT                      8) Configuration-Edit-Global-Settings
3) Application-View          9) Configuration-Consistency-Report
4) Configuration-Generate   10) Configuration-ScriptExecution
5) Configuration-Copy       11) RMS-ViewMachine
6) Configuration-Remove
Choose an action:
```

Figure 21: Main configuration menu when RMS is running

When RMS is running, the following entries either appear or change their behavior:

- *Application-View*—Views an existing application in read-only mode.
- *Configuration-Generate*—Same functionality as when RMS is not running.
- *Configuration-Copy*—Produces a copy of an existing configuration. This is often used to make a backup before an existing, tested configuration is enhanced.



Configuration-Copy cannot overwrite the configuration that is currently running.

- *Configuration-Remove*—Removes (deletes) any existing configuration except the one that is currently running.
- *RMS-ViewMachine*—Displays the list of nodes on which RMS is currently running.

4.3.3 Secondary menus

Each of the main menu items has a number of secondary menus. The secondary menus themselves can have sub-menus.

The *Creation: Application type selection menu* (Figure 22) is an example of a secondary menu. You see this menu after selecting *Application-Create* from the main menu.

```
Creation: Application type selection menu:
1) HELP                7) SCALABLE
2) QUIT                8) STANDBY
3) RETURN
4) OPTIONS
5) DEMO
6) GENERIC
Application Type: 8
```

Figure 22: Application type selection

4.3.4 Basic and non-basic settings

Basic and non-basic settings are designed to guide you safely through the configuration process, ensuring that all mandatory settings are configured.

Among the basic settings are the application name and the names of the nodes where it can run. For example, at the application type selection menu shown in the previous section, selecting 8) *STANDBY* produces the menu in Figure 23.

```
Consistency check ...
Yet to do: process the basic settings using Machines+Basics
Yet to do: process at least one of the non-basic settings

Settings of turnkey wizard "STANDBY" (APP1:not yet consistent)
1) HELP                4) REMOVE+EXIT
2) NO-SAVE+EXIT        5) ApplicationName=APP1
3) SAVE+EXIT           6) Machines+Basics(-)
Choose the setting to process: 6
```

Figure 23: Menu leading to basic settings

If you select 6) *Machines+Basics*, you can configure the basic settings using the menu in Figure 24.

```
Consistency check ...

Machines+Basics (app1:consistent)
1) HELP                                14) (AutoStartUp=no)
2) -                                    15) (AutoSwitchOver=No)
3) SAVE+EXIT                            16) (PreserveState=no)
4) REMOVE+EXIT                          17) (PersistentFault=0)
5) AdditionalMachine                    18) (ShutdownPriority=)
6) AdditionalConsole                    19) (OnlinePriority=)
7) Machines[0]=fuji2RMS                 20) (StandbyTransitions=)
8) (PreCheckScript=)                   21) (LicenseToKill=no)
9) (PreOnlineScript=)                  22) (AutoBreak=yes)
10) (PostOnlineScript=)                 23) (AutoBreakMaintMode=no)
11) (PreOfflineScript=)                 24) (HaltFlag=no)
12) (OfflineDoneScript=)                25) (PartialCluster=0)
13) (FaultScript=)                     26) (ScriptTimeout=)

Choose the setting to process:
```

Figure 24: Menu to configure basic settings

The menu displays the application's current attribute settings, some of which may be set automatically by the wizards. Attributes enclosed in parentheses are optional.

After you complete the configuration of the basic settings, the non-basic settings menu appears (Figure 24). Non-basic settings include specifications for resources such as file systems, IP addresses, disks, and so forth.

```
Consistency check ...
Yet to do: process at least one of the non-basic settings

Settings of turnkey wizard "STANDBY" (APP1:not yet consistent)
1) HELP                                9) Procedure:BasicApplication(-)
2) -                                    10) Symfoware(-)
3) SAVE+EXIT                            11) Procedure:SystemState3(-)
4) -                                    12) Procedure:SystemState2(-)
5) ApplicationName=APP1                 13) Gls:Global-Link-Services(-)
6) Machines+Basics(app1)               14) IpAddresses(-)
7) CommandLines(-)                     15) LocalFileSystems(-)
8) Procedure:Application(-)            16) Gds:Global-Disk-Services(-)

Choose the setting to process:
```

Figure 25: Menu to configure non-basic settings

i The list of available subapplications displayed in the menu depends on the packages installed on the local system. Some of the subapplications shown in this example may not be available in your market or for your platform.

4.4 Activating a configuration

As described in "General configuration procedure", activating a configuration is the third of the four fundamental steps required to set up a high-availability configuration. The activation phase comprises a number of tasks, among which are generation and distribution of a configuration.

i You must stop RMS on all the nodes in the cluster before you activate a configuration.

The starting point for the activation phase is the *Main configuration menu* (see Figure 26).

```

fuji2: Main configuration menu, current configuration: myconfig
No RMS active in the cluster
1) HELP                      10) Configuration-Remove
2) QUIT                       11) Configuration-Freeze
3) Application-Create         12) Configuration-Thaw
4) Application-Edit           13) Configuration-Edit-Global-Settings
5) Application-Remove         14) Configuration-Consistency-Report
6) Application-Clone          15) Configuration-ScriptExecution
7) Configuration-Generate     16) RMS-CreateMachine
8) Configuration-Activate     17) RMS-RemoveMachine
9) Configuration-Copy
Choose an action: 8

```

Figure 26: Main configuration menu

► Select the *Configuration-Activate* item by entering the number 8.

The activation is performed by the wizard. No further input is required at this stage.

During the activation phase, the wizard executes a series of tasks and displays the status on the screen. The completion of a task is indicated by the word *done* or a similar expression (see Figure 27).


```

About to activate the configuration myconfig ...

Testing for RMS to be up somewhere in the cluster ... done.
Arranging sub applications topologically ... done.
Check for all applications being consistent ... done.
Running overall consistency check ... done.
Generating pseudo code [one dot per (sub) application]: ..... done.
Generating RMS resources [one dot per resource]: ..... done
..... done

hybuild using /usr/opt/reliant/build/wizard.d/myconfig/myconfig.us
About to distribute the new configuration data to hosts: fuji2RMS,fuji3RMS

The new configuration was distributed successfully.

About to put the new configuration in effect ... done.

The activation has finished successfully.
Hit CR to continue

```

Figure 27: Activating a configuration

Among the tasks carried out by *Configuration-Activate* are generation and distribution of the configuration. The wizard performs a consistency check of the graph created in the generation of the configuration before distributing the configuration to all the nodes specified in the configuration.

The test to see whether RMS is up on one of the nodes in the cluster is required since activation cannot be performed if RMS is running. In this case, RMS would need to be shut down first.

i The *Configuration-Activate* process removes persistent status information on all affected nodes.

After the configuration has been activated successfully, you can return to the *Main configuration menu*. From there, you can quit the configuration procedure.

- ▶ Press **[Enter]** to return to the *Main configuration menu* (see Figure 28).

```
fuji2: Main configuration menu, current configuration: myconfig
No RMS active in the cluster
1) HELP                               10) Configuration-Remove
2) QUIT                               11) Configuration-Freeze
3) Application-Create                 12) Configuration-Thaw
4) Application-Edit                   13) Configuration-Edit-Global-Settings
5) Application-Remove                 14) Configuration-Consistency-Report
6) Application-Clone                  15) Configuration-ScriptExecution
7) Configuration-Generate              16) RMS-CreateMachine
8) Configuration-Activate              17) RMS-RemoveMachine
9) Configuration-Copy
Choose an action: 2
```

Figure 28: Quitting the Main configuration menu

- ▶ Select *QUIT* by entering the number 2.


This ends the activation phase of the configuration process. Usually, the next step is to start RMS to monitor the newly-configured application.

- ▶ Start RMS with the GUI or with the following command:

```
hvcn -a
```

4.5 Configuration elements

This section discusses some basic elements that are part of a high-availability configuration. Most of them have been mentioned in previous sections. Additional details are provided here to assist you in understanding how they are used by the wizards.

 Users do not have to deal with any of the items listed in this section directly. RMS Wizards manage all the basic elements for a high availability configuration. This section is provided only to help users better understand the configuration elements.


4.5.1 Scripts

Scripts are used in a high-availability configuration to perform several kinds of actions. Among the most important types of actions are the following:

- Bringing a resource to an `Online` state
- Bringing a resource to an `Offline` state

As an example of a script sending a resource `Offline`, you might think of a file system that has to be unmounted on a node where a fault occurs. An offline script would use the `umount` command to unmount the file system. Another script might use the `mount` command to mount it on a different node.

Besides such online and offline scripts, there are also pre-online and pre-offline scripts for preparing transition into the respective states, as well as a number of other scripts.

 The `hvexec` command executes scripts for a high-availability configuration monitored by RMS. For more details on the command `hvexec` please refer to the `primer.htm` document.

4.5.2 Detectors

Detectors are processes that have the task of monitoring resources. If there is a change in the state of a resource (for example, of a disk group) the detector in charge notifies the RMS base monitor. The base monitor may then decide to have a script executed as a reaction to this changed state.

4.5.3 RMS objects

A high-availability configuration can be seen as a set or group of objects with interdependencies. Any application or resource that is part of the configuration is then represented by one of the objects. The interdependences of objects can be displayed as a graph called the RMS graph.

These are the most important object types used in RMS configurations:

- `userApplication`—Represents an application to be configured for high-availability.
- `SysNode`—Represents a machine that is running as a node in a cluster.
- `gResource`—Represents a generic resource that is to be defined according to the needs of a customer application.
- `Controller`—Provides a dependency linkage so that a child application can act as a resource of the parent application.

In a typical configuration, one detector can be associated with all objects of the same type.

5 Example of configuration change

This chapter provides an example procedure of configuration change to pre-built cluster applications using the RMS Wizard with Solaris version PRIME-CLUSTER. The following below provides the example of configuration change.

- "Setting PreCheckScript in cluster application"
- "Changing timeout period of controller"



When editing the RMS configuration, make sure to stop RMS. Use the Cluster Admin GUI (See "Stopping RMS") or enter the following command from any machine in the cluster:

```
# hvshut -a
```



Note

When taking this procedure on Solaris, note the following things:

- Before taking this procedure, complete all the settings of the cluster application first on the userApplication Configuration Wizard.
- Use the `wgcnfclient` command to check the RMS Configuration name specified by `-n` option when executing the `hvw` command. For details on how to use the `wgcnfclient` command, see "6.7.6 Changing the RMS Configuration Name" in "PRIMECLUSTER Installation and Administration Guide (Oracle Solaris)."
- On Solaris, make sure to execute the `hvw` command with `-xj` option.

5.1 Setting PreCheckScript in cluster application

See the procedure below to set the `PreCheckScript` in the cluster application.



Note

When adding/deleting a resource or changing the setting of a resource in the cluster application on Solaris, delete the `userApplication` once as a setting procedure. In this case, follow this procedure to set the `PreCheckScript` again.

- ▶ Enter the following command to display the wizard menu for configuring RMS.

```
# hvw -xj -n configname
```

- ▶ RMS Wizard is started and the Main configuration menu is displayed.

Enter "Application-Edit" and then select *Application-Edit*.

```
crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJ$VWVUCWGUIPROMPT:
FJ$VWVUCWGUIPROMPT:
Application-Edit
```

Figure 29: Main configuration menu

- ▶ Enter "OPTIONS" and then select *OPTIONS*.

Example of configuration change **Setting PreCheckScript** in cluster applicati

```
Edit: Application selection menu (restricted):
HELP
QUIT
RETURN
OPTIONS
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
OPTIONS
```

Figure 30: Application selection menu

- ▶ Enter "ShowAllAvailableWizards" and then select *ShowAllAvailableWizards*.

```
Edit: selection criteria:
HELP
RETURN
NONE
ShowTurnkeyWizardsOnly
ShowAllAvailableWizards
ApplicationsOnly
SearchPattern
ShowSubApplications
FlagApplications
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
ShowAllAvailableWizards
```

Figure 31: Option menu of the application setting

- ▶ In the screen below, enter the application name that sets the PreCheckScript, and enter the return key.

Here, enter "userApp_0" to set PreCheckScript of *userApp_0*.

```
Edit: Application selection menu:
HELP
QUIT
RETURN
OPTIONS
Cmdline0
userApp_0
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
userApp_0
```

Figure 32: Application selection menu

Example of configuration change **Setting PreCheckScript** in cluster applicati

Enter "PreCheckScript" and then select (*PreCheckScript=*).

```
Settings of application type "generic" (consistent)
HELP
NO-SAVE+EXIT
SAVE+EXIT
ApplicationName=userApp_0
AdditionalMachine
AdditionalConsole
AdditionalSubApplication
Machines[0]=crowRMS
SubApplications[0]=Cmdline0
(HostSpecificSubApplication=no)
(PreCheckScript=)
(PreOnlineScript=)
(PostOnlineScript=)
(PreOfflineScript=)
(PostOfflineScript=)
(OfflineDoneScript=)
(FaultScript=)
(AutoStartUp=no)
(AutoSwitchOver=HostFailure|ResourceFailure|ShutDown)
(PreserveState=no)
(PersistentFault=0)
(ShutdownPriority=)
(OnlinePriority=0)
(StandbyTransitions=No)
(LicenseToKill=no)
(AutoBreak=yes)
(AutoBreakMaintMode=no)
(HaltFlag=no)
(PartialCluster=0)
(ScriptTimeout=)
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
PreCheckScript
```

Figure 33: Application setting menu

► Enter "FREECHOICE" and then select *FREECHOICE*.

```
HELP
RETURN
NONE
FREECHOICE
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
FREECHOICE
```

Figure 34: Selection menu of the PreCheckScript

Example of configuration change **Setting PreCheckScript** in cluster applicati

- ▶ Enter the file path of the `PreCheckScript`, and press the return key.

In the screen below, `/usr/local/app/PreCheck.sh` is specified for the `PreCheckScript`.

```
HELP
RETURN
NONE
FREECHOICE
F.JSVWVUCWGUIPROMPT:
F.JSVWVUCWGUIPROMPT:
FREECHOICE
F.JSVWVUCWGUIPROMPT:
F.JSVWVUCWGUIPROMPT:
/usr/local/app/PreCheck.sh
```

Figure 35: Entering `PreCheckScript`

- ▶ Enter "SAVE+EXIT" and then select `SAVE+EXIT`.

```
Settings of application type "generic" (consistent)
HELP
NO-SAVE+EXIT
SAVE+EXIT
ApplicationName=userApp_0
AdditionalMachine
AdditionalConsole
AdditionalSubApplication
Machines[0]=crowRMS
SubApplications[0]=Cmdline0
(HostSpecificSubApplication=no)
(PreCheckScript='/usr/local/app/PreCheck.sh')
(PreOnlineScript=)
(PostOnlineScript=)
(PreOfflineScript=)
(PostOfflineScript=)
(OfflineDoneScript=)
(FaultScript=)
(AutoStartUp=no)
(AutoSwitchOver=HostFailure|ResourceFailure|ShutDown)
(PreserveState=no)
(PersistentFault=0)
(ShutdownPriority=)
(OnlinePriority=0)
(StandbyTransitions=No)
(LicenseToKill=no)
(AutoBreak=yes)
(AutoBreakMaintMode=no)
(HaltFlag=no)
(PartialCluster=0)
(ScriptTimeout=)
F.JSVWVUCWGUIPROMPT:
F.JSVWVUCWGUIPROMPT:
SAVE+EXIT
```

Figure 36: Application setting menu

- ▶ Enter "RETURN" and then select `RETURN`.

Example of configuration change **Setting PreCheckScript** in cluster applicati

```
Edit: Application selection menu:
HELP
QUIT
RETURN
OPTIONS
CmdLine0
userApp_0
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
RETURN
```

Figure 37: Application selection menu

- ▶ Enter "Configuration-Activate" and then select *Configuration-Activate*.

```
crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
Configuration-Activate
```

Figure 38: Main configuration menu

- ▶ RMS configuration is generated and distributed.

```

About to activate the configuration config ...

Testing for RMS to be up somewhere in the cluster ... done.
Arranging sub applications topologically ... done.
Check for all applications being consistent ... done.
Running overall consistency check ... done.
Generating pseudo code [one dot per (sub) application]: .... done.
Generating RMS resources..... done

hvbuild using /usr/opt/reliant/build/wizard.d/config/config.us
About to distribute the new configuration data to hosts: crowdRMS
The new configuration was distributed successfully.
About to put the new configuration in effect ... done.
The activation has finished successfully.

```

Figure 39: Generation and distribution of the RMS configuration file

- ▶ Enter "QUIT" and then select *QUIT*.

```

crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
QUIT

```

Figure 40: Quitting main configuration menu

5.2 Changing timeout period of controller

- ▶ Enter the following command to display the wizard menu for configuring RMS.

```
# hvw -xj -n configname
```

- ▶ Enter "Application-Edit" and then select *Application-Edit*.

```
crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
Application-Edit
```

Figure 41: Main configuration menu

- ▶ Enter "OPTIONS" and then select *OPTIONS*.

```
Edit: Application selection menu (restricted):
HELP
QUIT
RETURN
OPTIONS
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
OPTIONS
```

Figure 42: Application selection menu

- ▶ Enter "ShowAllAvailableWizards" and then select *ShowAllAvailableWizards*.

```
Edit: selection criteria:
HELP
RETURN
NONE
ShowTurnkeyWizardsOnly
ShowAllAvailableWizards
ApplicationsOnly
SearchPattern
ShowSubApplications
FlagApplications
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
ShowAllAvailableWizards
```

Figure 43: Option menu of application setting

- ▶ In the screen below, enter the resource name for the *Controller* resource that you want to change the timeout period and then press the return key.

Enter "ScalableCtrl_0" to change the timeout period of *ScalableCtrl_0*.

```
Edit: Application selection menu:
HELP
QUIT
RETURN
OPTIONS
CmdLine0
ScalableCtrl_0
userApp_0
userApp_1
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
ScalableCtrl_0
```

Figure 44: Application selection menu

- ▶ Enter "Controllers" and then select *Controllers*.

```

Settings of application type "Controller" (consistent)
HELP
NO-SAVE+EXIT
SAVE+EXIT
ApplicationName=ScalableCtrl_0
ControlPolicy=SCALABLE
AdditionalAppToControl
Controllers[0]=T3600:userApp_0
(FaultScript=)
(ApplicationSequence=userApp_0)
(StateChangeScript=)
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
Controllers

```

Figure 45: Settings menu of controller

- ▶ Enter "SELECTED:<userApplication name>" and then select *SELECTED*.

```

HELP
RETURN
NONE
FREECHOICE
SELECTED:userApp_0
userApp_1
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
SELECTED

```

Figure 46: Selection menu of controlled target application

- ▶ Enter "TIMEOUT" and then select *TIMEOUT(T)*.

```

Set *global* flags for all scalable (sub) applications: SELECTED
Currently set: TIMEOUT (T3600)
HELP

SAVE+RETURN
DEFAULT
MONITORONLY(M)
TIMEOUT(T)
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
TIMEOUT

```

Figure 47: Attribution setting menu of the controller

- ▶ Enter "FREECHOICE" then select *FREECHOICE* and then enter the timeout period. Here, enter "2340" to set the timeout period to 2340 seconds.

```

HELP
RETURN
FREECHOICE
180
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
FREECHOICE
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
2340

```

Figure 48: TIMEOUT input

- ▶ Enter "SAVE+RETURN" and then select *SAVE+RETURN*.

```

Set *global* flags for all scalable (sub) applications: SELECTED
Currently set: TIMEOUT (T2340)
HELP

SAVE+RETURN
DEFAULT
MONITORONLY(M)
TIMEOUT(T)
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
SAVE+RETURN

```

Figure 49: Attribution setting menu of the controller

- ▶ Enter "SAVE+EXIT" and then select *SAVE+EXIT*.

```

Settings of application type "Controller" (consistent)
HELP
NO-SAVE+EXIT
SAVE+EXIT
ApplicationName=ScalableCtrl_0
ControlPolicy=SCALABLE
AdditionalAppToControl
Controllers[0]=T2340:userApp_0
(FaultScript=)
(ApplicationSequence=userApp_0)
(StateChangeScript=)
FJSVWVUCWGUIPROMPT:
FJSVWVUCWGUIPROMPT:
SAVE+EXIT

```

Figure 50: Menu settings of the controller

- ▶ Enter "RETURN" and then select *RETURN*.

```
Edit: Application selection menu:
HELP
QUIT
RETURN
OPTIONS
Cmdline0
ScalableCtrl_0
userApp_0
userApp_1
FJSVWVUCNGUIPROMPT:
FJSVWVUCNGUIPROMPT:
RETURN
```

Figure 51: Application selection menu

- ▶ Enter "Configuration-Activate" and then select *Configuration-Activate*.

```
crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJSVWVUCNGUIPROMPT:
FJSVWVUCNGUIPROMPT:
Configuration-Activate
```

Figure 52: Main configuration menu

- RMS configuration is generated and distributed.

```

About to activate the configuration config ...

Testing for RMS to be up somewhere in the cluster ... done.

Arranging sub applications topologically ... done.

Check for all applications being consistent ... done.

Running overall consistency check ... done.

Generating pseudo code [one dot per (sub) application]: .... done.

Generating RMS resources..... done

hybuild using /usr/opt/reliant/build/wizard.d/config/config.us
About to distribute the new configuration data to hosts: crowdRMS

The new configuration was distributed successfully.

About to put the new configuration in effect ... done.

The activation has finished successfully.

```

Figure 53: Creation and distribution of designated file of RMS configuration

- Enter "QUIT" and then select *QUIT*.

```

crow: Main configuration menu, current configuration: config
No RMS active in the cluster
HELP
QUIT
Application-Create
Application-Edit
Application-Remove
Application-Clone
Configuration-Generate
Configuration-Activate
Configuration-Copy
Configuration-Remove
Configuration-Freeze
Configuration-Thaw
Configuration-Edit-Global-Settings
Configuration-Consistency-Report
Configuration-ScriptExecution
RMS-CreateMachine
RMS-RemoveMachine
FJSVWUCWGUIPROMPT:
FJSVWUCWGUIPROMPT:
QUIT

```

Figure 54: Quitting main configuration menu

6 Using the Cluster Admin GUI

This chapter describes PRIMECLUSTER administration using the Cluster Admin graphical user interface (GUI). In addition, some command-line interface (CLI) commands are discussed.

Chapter contents:

- "Overview"
- "Starting the Cluster Admin GUI"
- "Monitoring RMS with Cluster Admin"

6.1 Overview

RMS administration can be done by means of the Cluster Admin GUI or by the CLI. The procedures in this chapter focus on the Cluster Admin GUI.

CLI procedures

Some of the operations described in this chapter describe the equivalent CLI procedure. However, we recommend that the CLI be used only by expert system administrators, or in those cases where a browser is not available. If you decide to use a CLI procedure, please note the following:

- The commands are located in the `<RELIANT_PATH>/bin` directory.
- All RMS CLI commands accept both CF node names and RMS node names for `SysNode` objects when the RMS naming convention has been followed (that is, when the names are of the form `<CF-name>RMS`).
- RMS CLI commands accept case-insensitive application names.
- The CLI procedures presented here are not intended to be a complete description of the commands that are used. Other options may be available. For more information about any CLI command, see its online manual pages. For the complete list of online documentation related to RMS operation, see "Appendix—Release information".

CLI status codes

In general, RMS `hv*` commands send a request to the base monitor and then return immediately without waiting for this request to be processed. They exit with a status code of 0 (success), which indicates a request has been **sent** to the base monitor successfully. However, this does not guarantee that the request was **processed** successfully.

Important exceptions include the `‘hvshut’` and `‘hvutil -[mM]’` commands, which remove nodes and applications from RMS control. These commands may return status codes that indicate failure. This type of command also provides an option for “forced” operation, but this should be used with great care and only when absolutely necessary.

6.2 Starting the Cluster Admin GUI

The following sections discuss how to use the RMS portion of the GUI.



Windows desktop systems require the Java plugin as specified in "PRIMECLUSTER Web-Based Admin View Operation Guide" for your operating system.

6.2.1 Web-Based Admin View

- Open a Java-enabled browser and enter the following URL in the *Address* location:

```
http://<hostname>:8081/Plugin.cgi
```

The *hostname* should be the name or IP address of the primary or secondary management server. For example, if a cluster has `shasta1` and `shasta2` as its primary and secondary management servers, the URL would be either one of the following:

- `http://shasta1:8081/Plugin.cgi`
- `http://shasta2:8081/Plugin.cgi`

The `Plugin.cgi` URL always attempts to contact the primary management server; after contacting the host, the browser changes the URL suffix from `‘.cgi’` to `‘.html’`. If you instead use the `Plugin.html` form, Cluster Admin will attempt to make direct contact with the server specified in the URL.

For details on the primary and secondary management servers, refer to "PRIMECLUSTER Web-Based Admin View Operation Guide" for your operating system.

6.2.2 Login

Before logging in, make sure you have a user name and password with the appropriate privilege level. Cluster Admin has the following privilege levels:

- Root privileges—Can perform all actions including configuration, administration, and viewing tasks.
- Administrative privileges—Can view and execute commands, but cannot make configuration changes.
- Operator privileges—Can only perform viewing tasks.

For more details on privilege levels, refer to "PRIMECLUSTER Installation and Administration Guide" for your operating system.

After the Web-Based Admin View login window appears (Figure 55), log in as follows:

- ▶ Enter the user name and password for a user with the appropriate privilege level.
- ▶ Click the *OK* button.

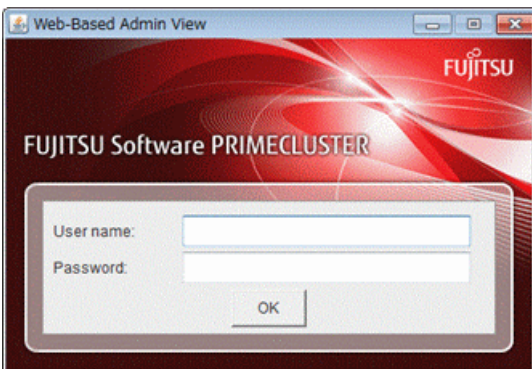


Figure 55: Web-Based Admin View login

After you log in, the Web-Based Admin View window appears (Figure 56).

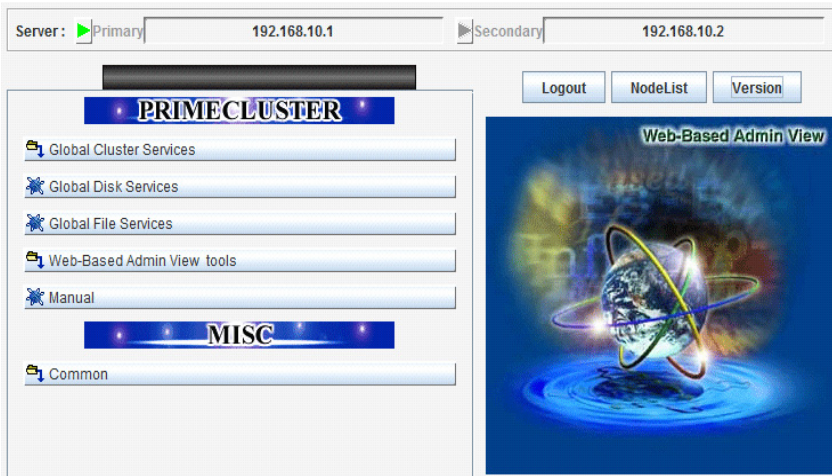


Figure 56: Invoking the Cluster Services GUI

- ▶ Click the *Global Cluster Services* button to advance to the next view (Figure 57).

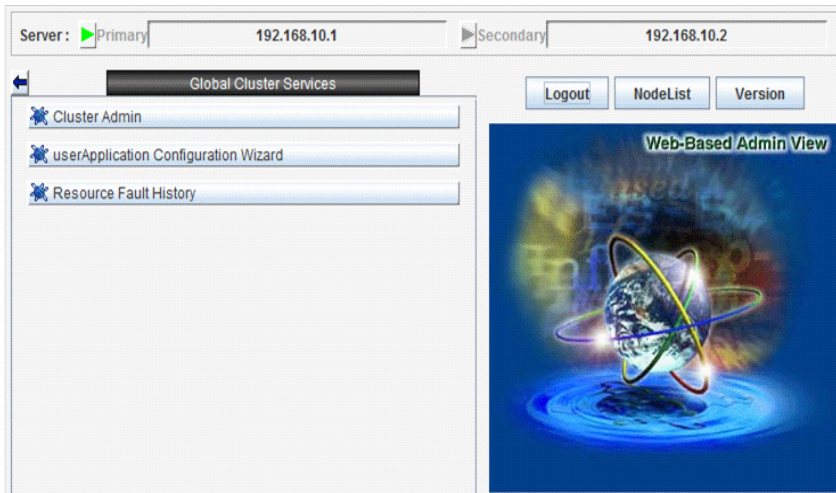


Figure 57: Invoking Cluster Admin

- ▶ Click the *Cluster Admin* button.

The *Choose a node for initial connection* window appears (Figure 58).

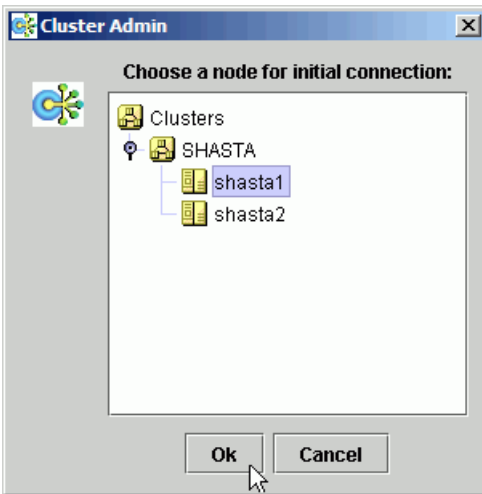


Figure 58: Cluster Admin initial connection menu

The nodes are displayed in alphabetical order, and the first one is selected by default. In most cases, the node you choose is immaterial for administrative tasks.

- ▶ Select the desired node for the connection, and click *OK*.

The next window that appears depends on how you have set the trust levels for the Cluster Admin Java applets. If you have already chosen to use trusted applets for all sessions, you can skip the following description.

Trusted applets

For platform independence and security, the Cluster Admin GUI uses Java applets. When the Java applets run in trusted mode, they are allowed to use some client system resources, such as the clipboard. If you intend to copy and paste text between the Java window and other applications on your workstation, you must run the applets in trusted mode.

6.2.3 Main Cluster Admin window

When Cluster Admin opens, the initial view is similar to Figure 59.

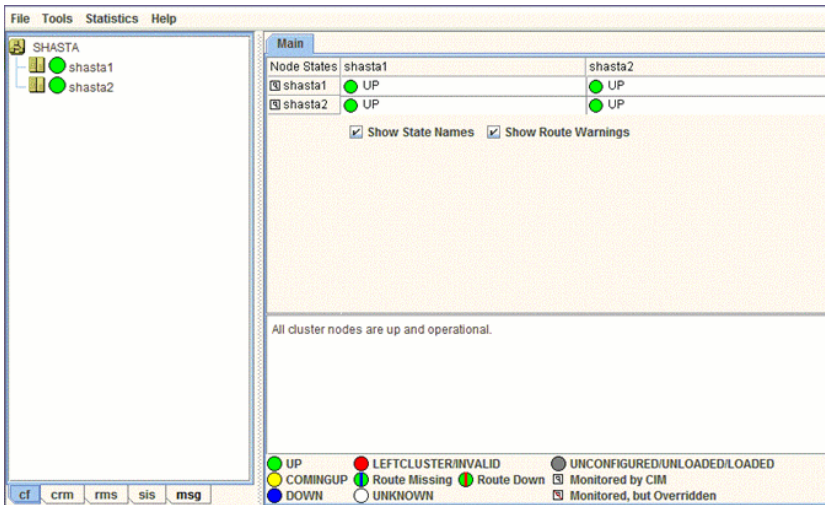


Figure 59: Main Cluster Admin window—Initial view

The following tabs appear at the bottom of the left pane:

- *cf*
- *rms*
- *sis*
- *msg* (message window)

Clicking a tab switches the view to the corresponding product. Initially, the CF view is selected. All views have the following common features:

- **Menu bar**—The pull-down menus contain generic administrative functions as well as items specific to the PRIMECLUSTER products. Some entries may be disabled (grayed out) according to the item selected in the current view.
- **Configuration tree**—The left pane contains product-specific configuration information in a hierarchical display. Clicking on an item in the tree will display the item's properties and, in some views, allow you to change them. Right-clicking on an item generally brings up a context menu specific to that item.
- **Input and message area**—The large pane on the right is the main work and information area. The content varies according to the product being administered and the functions selected from the menus or tree.

This chapter focuses on RMS administration. For information about operations available on the other tabs, refer to the corresponding product documentation.

**Note**

SIS is not available in PRIMECLUSTER 4.5.

6.2.4 Cluster Admin message view

Error and debug messages related to Cluster Admin can be displayed at any time:

- ▶ Click the *msg* tab on the bottom of the RMS tree pane. (The tab label is red if a new message has been added to the text area since it was last viewed.) The *Admin Errors and Messages* view appears (Figure 60).

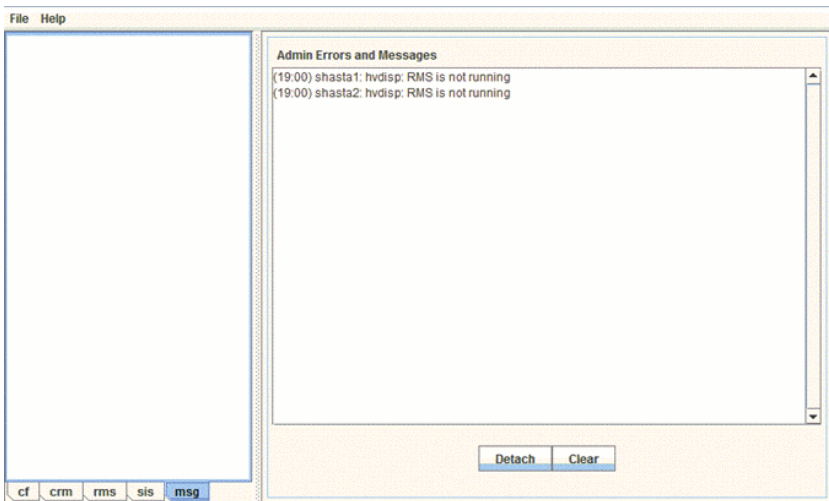


Figure 60: Main Cluster Admin window—message view

The message pane can be detached or re-attached using the buttons at the bottom of the display. Use the *Clear* button to delete all messages in the display.

6.3 Monitoring RMS with Cluster Admin

The procedures in this section allow you to view information about the RMS cluster as well as individual nodes, applications, and resources. These procedures are passive: they display data, but they do not change the operation of the configuration.

- ▶ To enter the RMS portion of the Cluster Admin GUI, click on the *rms* tab.

A typical RMS view is shown in Figure 61.

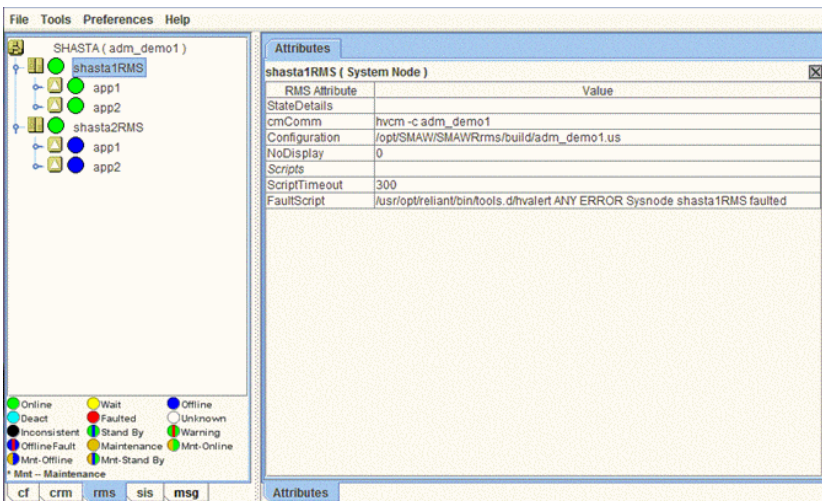


Figure 61: Main Cluster Admin window—RMS view

The main window area is split into two major areas: the left pane contains the RMS tree; the right pane displays configuration information, properties of nodes and objects, RMS logs, or other items. The information displayed depends on what has been selected RMS tree and which operation, if any, has been invoked.

6.3.1 RMS tree

The Cluster Admin RMS tree displays the configuration information for the cluster in a simple hierarchical format. The tree has the following levels:

- Root of the tree—Represents the cluster. The root is labeled with the cluster name, followed by the RMS configuration name in parentheses. The cluster name is determined by your CF configuration.
- First level—Represents the system nodes forming the cluster.
- Second level—Represents the `userApplication` objects running on each of the system nodes.
- Third level—Represents subapplications, if any. Also contains non-affiliated groups of objects (see fourth level description).
- Fourth level—Represents the resources necessary for each of the subapplications. Also contains non-affiliated objects.



Non-affiliated `andOP` and `orOP` objects are intended for use by RMS experts. These items provide logical dependencies and group connectivity between nodes, applications, and subapplications.

If an application has subapplications, the fourth level represents resources used by that subapplication. If an application does not have subapplications, then the third level represents all the resources used by the `userApplication`.

Applications always appear at the second level of the tree, even if some of them are controlled by others. Dependencies between applications are depicted in the RMS tree by the presence of controller objects. An example of an RMS tree with a controller object is shown in Figure 62.

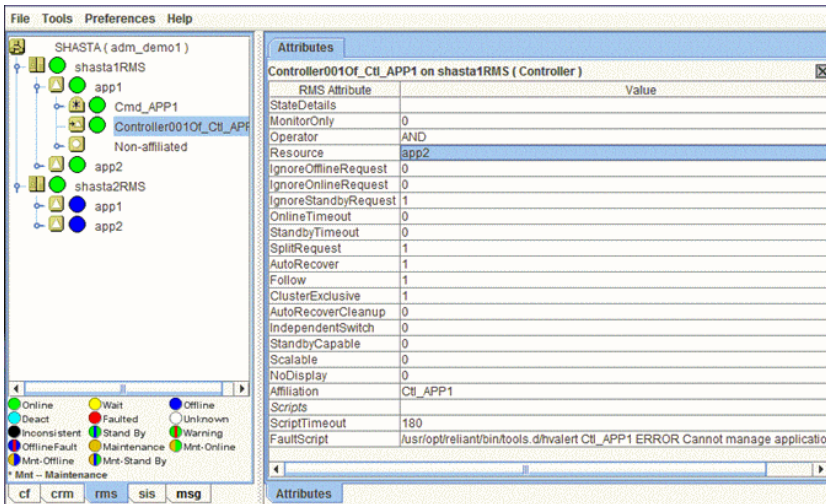


Figure 62: RMS tree with a controller object

In the example shown in the figure, the *Resource* attribute of the highlighted controller object in *app1* contains the name of the *app2* application: this indicates the dependence of *app1* on *app2*.

A complete visual display of object dependence is available from the RMS graph. See the "Using RMS graphs" for a description of the procedure.

6.3.2 Pop-up context menus

Each object in the Cluster Admin RMS configuration tree has a pop-up context menu that provides quick access to commonly-used operations. Invoke the context menu by right-clicking on any object. The first item on the menu displays the selected object's name in grayed-out text. The remaining items list the available operations, which vary according to the object's type and current state (Figure 63).

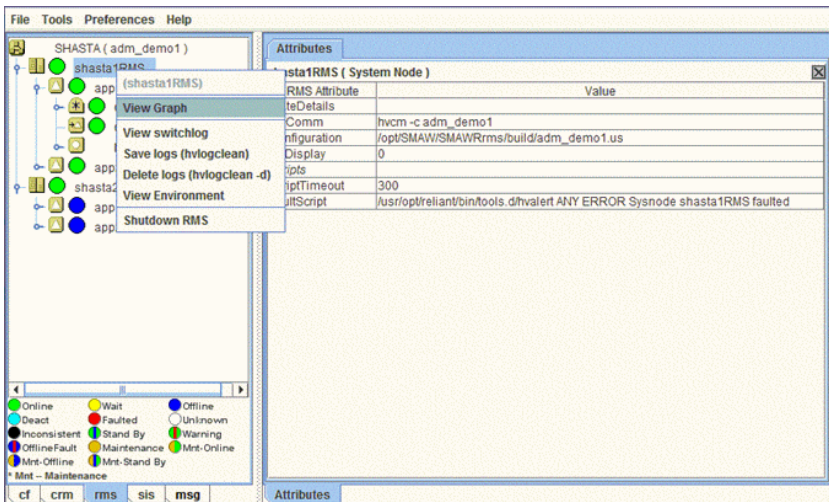


Figure 63: Pop-up context menu for a node

Items that affect object states, node states, or the entire RMS configuration appear toward the bottom of the menu. These actions are described in the final sections of this chapter.



To close the context menu without performing an operation, click the grayed-out object name in the menu or press the **[Esc]** key.

The menu offers different operations for a node object compared to an application object. It also offers different options for an application object in the online state (Figure 64) compared to the offline state (Figure 65).

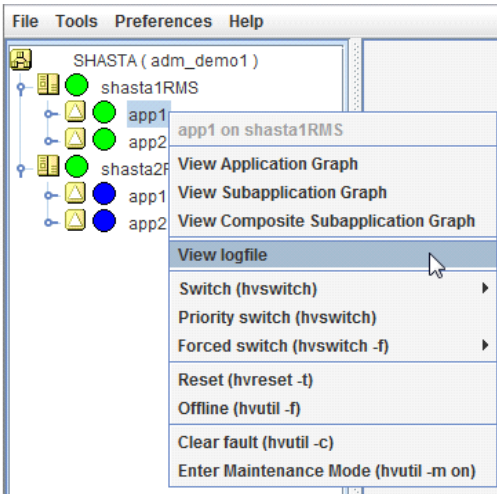


Figure 64: Pop-up context menu for an online application

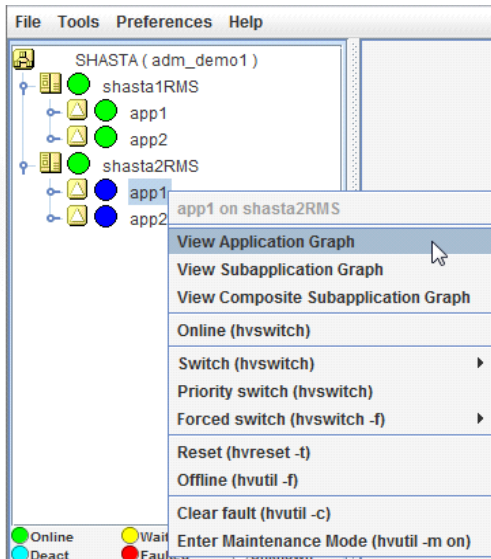


Figure 65: Pop-up context menu for an offline application

6.3.3 Pop-up confirmation dialogs

When you select an item in an object's context menu that can cause state changes to that object, a pop-up confirmation dialog appears (Figure 66). To proceed with the action described in the warning message, click *Yes*; to cancel the action, click *No*.

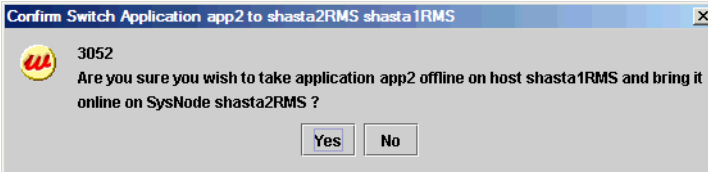


Figure 66: Pop-up confirmation dialog

6.3.4 Displaying environment variables

Display the global (clusterwide) environment variables as follows:

- ▶ Right-click on a cluster in the RMS tree window and select *View Environment* from the context menu (Figure 67).

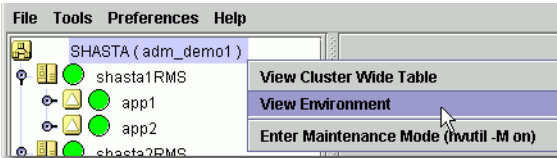


Figure 67: Displaying global environment variables

The global variables will appear under a separate tab in the right pane (Figure 68).

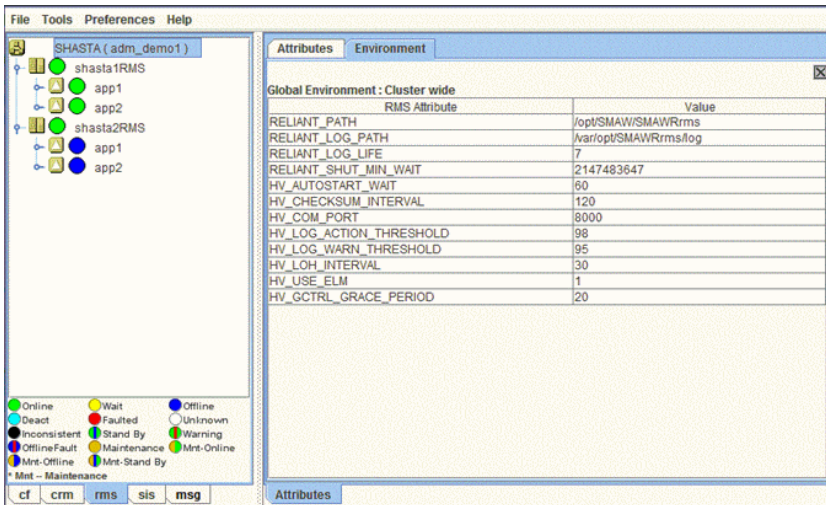


Figure 68: Global environment variable view



To close a tab view, use the button in the upper-right corner of the view.

Display local environment variables as follows:

- ▶ Right-click on a node in the RMS tree window and select *View Environment* from the context menu (Figure 69).

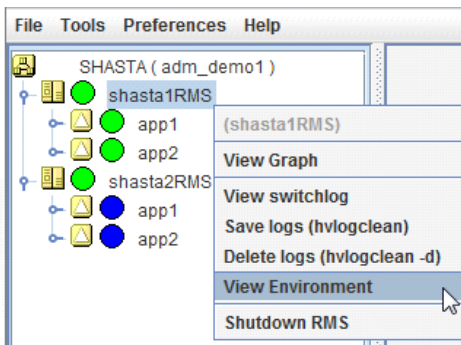


Figure 69: Displaying local environment variables

Both local and global variables appear on the same tab in the right pane (Figure 70).

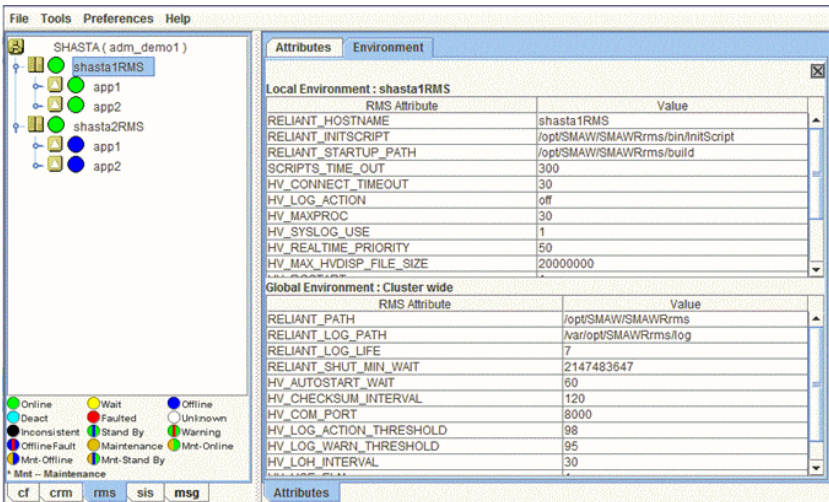


Figure 70: Local environment variables view

CLI: hvdisp

Display the environment variables with the `hvdisp` command, which does not require root privilege:

```
hvdisp ENV
hvdisp ENVL
```

6.3.5 Displaying object states

The state of each RMS object is indicated by the color of its circular status icon, located immediately to the left of the object's name in the configuration tree. The legend for the object states appears below the RMS tree in the left pane of the RMS view (Figure 71).

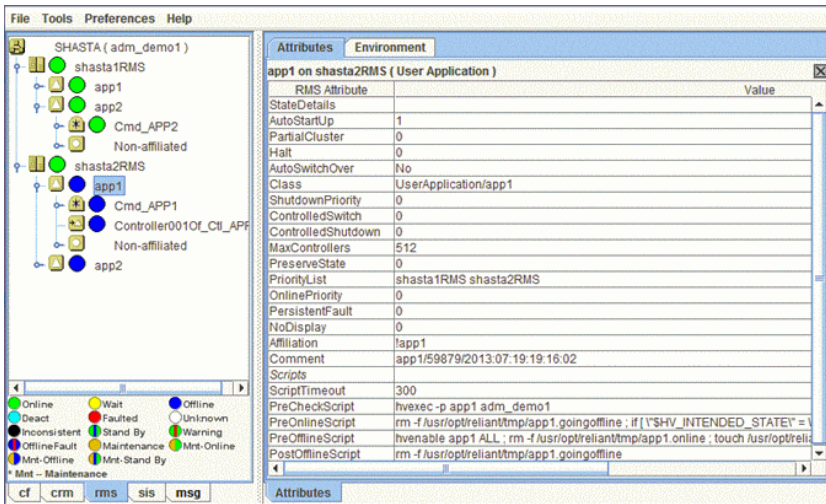


Figure 71: Displaying application and object states

In the example above, the application `app2` is online (green status icon) on node `shasta1RMS`, but offline (blue status icon) on node `shasta2RMS`.

See "Appendix—States" for a description of RMS object states.

CLI: `hvdisp`

The syntax for the CLI is as follows:

```
hvdisp {-a | -c} [-o out_file]
```

Options:

- a Displays the object name, the object type, the object's `SysNode` name, and the object state for each object in the configuration (automatically generated connectors are not shown)
- c Displays information in compact format
- o Sends the output to the designated file

The `hvdisp` command only works when RMS is running and does not require root privilege.

6.3.6 Configuration information or object attributes

View configuration information for individual objects by left-clicking with the mouse on the object in the tree. The properties are displayed in a tabular format on the right pane of the RMS main window (Figure 72).

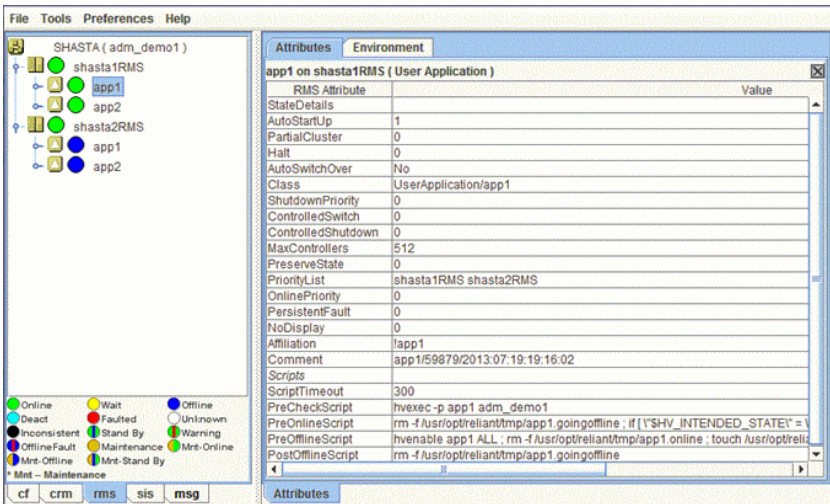


Figure 72: Configuration information or object attributes

7 Additional administrative tools

The Cluster Admin GUI provides several additional tools to help you manage RMS operation.

Chapter contents:

- "Using the RMS clusterwide table"
- "Using RMS graphs"
- "Interpreting display changes"
- "Viewing RMS log messages"

7.1 Using the RMS clusterwide table

The RMS clusterwide table displays the state of each application on each of the system nodes in a concise table.

- ▶ To open the clusterwide table, right-click the cluster name (the root of the RMS tree in the left pane) and then select *View Cluster Wide table* from the context menu (Figure 73).

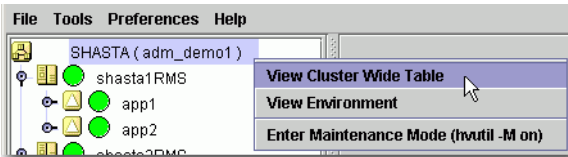


Figure 73: Opening the clusterwide table

The clusterwide table appears in a separate window (Figure 74).

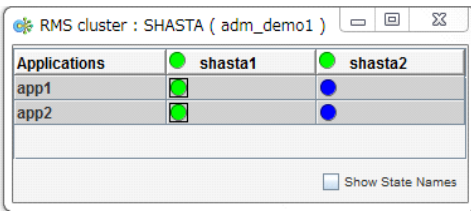


Figure 74: Clusterwide table

- To display the corresponding state name next to each status icon, click the *Show State Names* checkbox at the lower-right corner of the window (Figure 75).

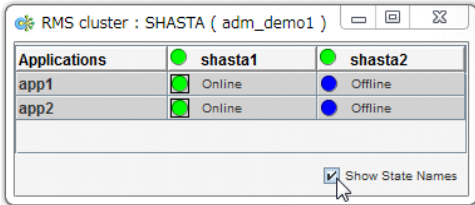


Figure 75: Clusterwide table with state names

You can increase or decrease the size of the clusterwide table window and the size of the columns by using the mouse. If the window is already large enough to fully display all of the table elements, then you will not be allowed to further increase its size.

A square surrounding the colored state circle indicates the primary node for the application. Figure 75 above shows that `shasta1` is the primary node for all of the applications.

Normally, the clusterwide table displays applications in alphabetical order from top to bottom. However, `Faulted` applications are handled specially. If an application is in the `Faulted` state on any node in the cluster, then it is displayed at the top of the table, and the application's name is highlighted by a pink background (Figure 76). This allows the System Administrator to easily spot any `Faulted` applications.

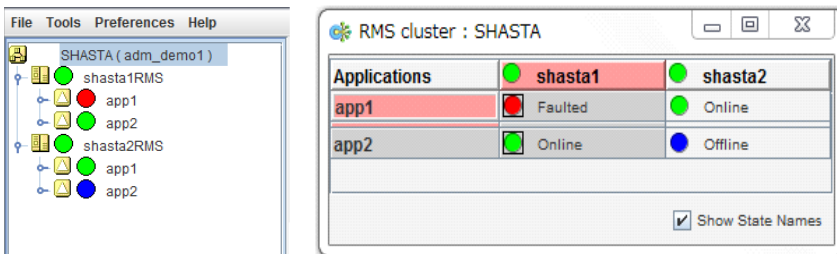


Figure 76: Faulted applications in the clusterwide table

The clusterwide table also makes special provisions for applications that are not online anywhere in the cluster. These applications are also displayed at the top of the table, with the application's name highlighted in light blue (Figure 77). This alerts the system administrator that some applications are not running anywhere and should probably be brought online on some node.

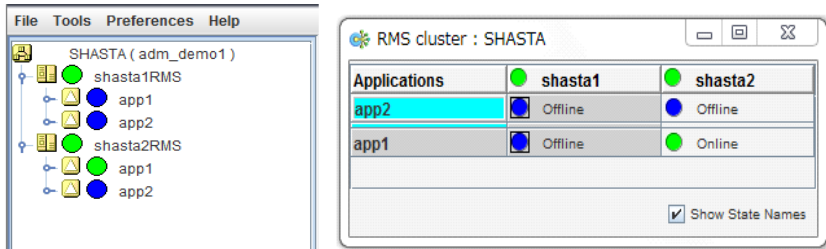


Figure 77: Offline applications in the clusterwide table

If there are both faulted applications and applications that are not online anywhere, then the faulted applications are listed first (Figure 78).

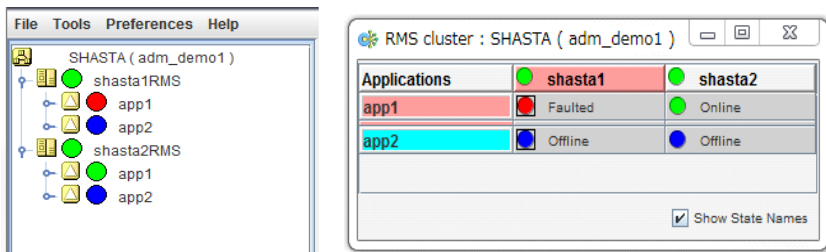


Figure 78: Faulted and offline applications in the clusterwide table

If there is a split-brain condition in the cluster on both the clusterwide table and the RMS tree, then colored exclamation marks will appear after the status icons (colored circles) of the nodes. A colored exclamation mark indicates that the state of that SysNode is different from what another SysNode views it as being. The color of the exclamation mark indicates the state that the other node thinks that the SysNode is in. If there are multiple nodes that see a SysNode in different states, you will see multiple exclamation marks after the colored circle. Exclamation marks are sorted according to the severity of the states.

Figure 79 shows a clusterwide table with an application of a split-brain condition. Note the yellow exclamation mark before the second node name.

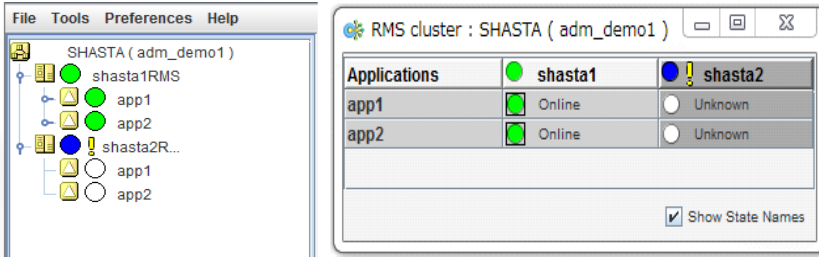


Figure 79: Split-brain conditions in the clusterwide table



Momentary split-brain conditions may be indicated while a node starts up or shuts down.

7.1.1 Using context menus from the clusterwide table

Each object in the clusterwide table has a pop-up context menu that provides quick access to commonly-used operations.

- ▶ To display a context menu, right-click on any object with a status icon, where column headers represent hosts and cells represent applications.

The first item on the menu displays the selected object’s name in grayed-out text. The remaining items list the available operations, which vary according to the object’s type and current state (Figure 80).

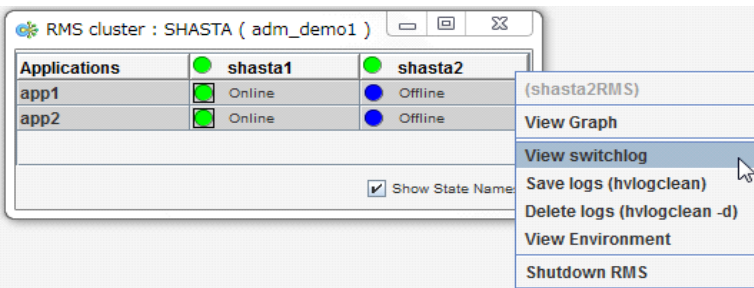


Figure 80: Using context menus in the clusterwide table

Items that affect object states, node states, or the entire RMS configuration appear toward the bottom of the menu. These actions are described in the final sections of this chapter.

- ▶ To close the context menu without performing an operation, click the grayed-out object name in the menu or press the `Esc` key.

7.2 Using RMS graphs

Cluster Admin provides an alternate way of viewing the RMS configuration hierarchy called **graphs**. A graph represents the configuration in a true tree structure, where the branches indicate the dependencies that are not generally visible in the RMS configuration tree described earlier. The following types of graphs are available:

- Full graph—Displays the complete cluster configuration.
- Application graph—Shows all of the resources used by an application and can be used to look at specific resource properties.
- Subapplication graph—Lists all of the subapplications used by a given application, and it shows the connections between the subapplications.
- Composite subapplications graph—Shows all the subapplications that the application depends on directly or indirectly.

The following sections describe each type in more detail, as well as these graph-related features:

- Obtaining configuration information
- Using command context (pop-up) menus
- Displaying various levels of detail
- Interpreting display changes in the clusterwide table and graphs

7.2.1 RMS full graph

- ▶ To display the RMS full graph, right-click on any system node and select *View Graph* from the context menu (Figure 81).

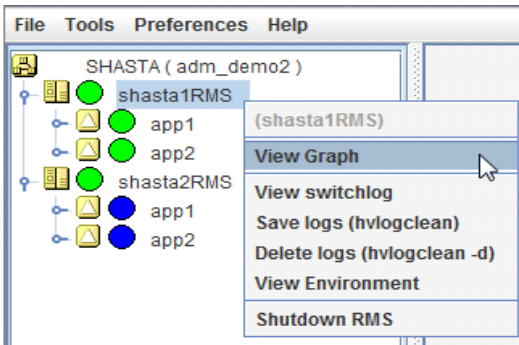


Figure 81: Viewing the RMS full graph on a node



The *View Graph* menu item is not available if a graph is already open for that node.

By default, each graph appears as a separate tab in the right pane of the Cluster Admin view (Figure 82).

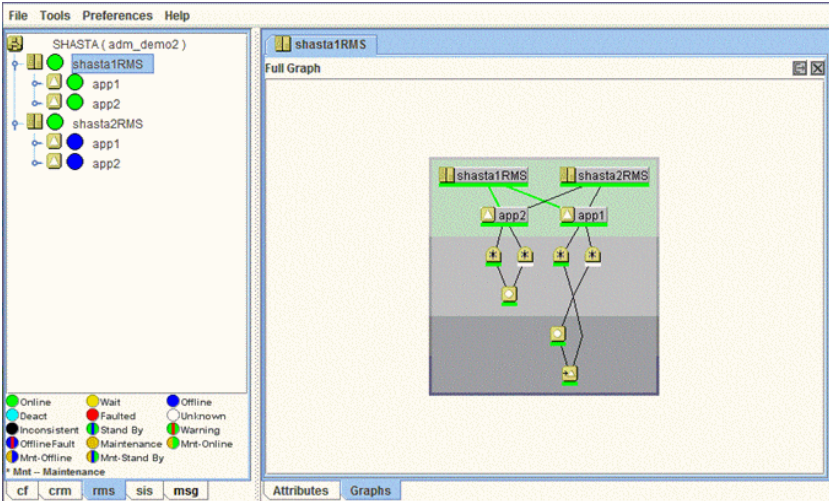


Figure 82: RMS full graph on a node—tab view

- ▶ To view any tab in a separate window, click the **detach** control button. The detach button is located next to the **close** control button in the upper-right corner of the view (Figure 83).



Figure 83: Detail of tab view showing detach button

The detached view contains the same information as the tabbed view (Figure 84).

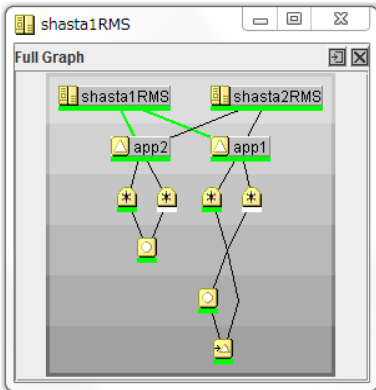


Figure 84: RMS full graph on a node—separate window view

- ▶ To rejoin the detached window to the Cluster Admin view, click the **attach** control button. The attach button is located next to the view's **close** control button in the upper-right corner, just below the standard window control buttons (Figure 85).

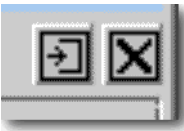


Figure 85: Detail of separate window view showing attach button

The RMS full graph displays the complete RMS configuration of the cluster and represents the following items:

- Node where each application is currently online, indicated by green lines between the node and application objects
- Object types, indicated by the object's icon
- Current object state, indicated by the colored bar beneath each icon
- Relationships between objects
- Dependencies of objects

The RMS graph is drawn from the perspective of the selected node; that is, the state information of all other objects is displayed according to the reports received by that node. The node name in the title bar of the graph identifies the node that is supplying the state information. You can create an RMS graph from the perspective of any node in the cluster.

The background of the graph is shaded from top to bottom with progressively darker gray bands. In large, complicated graphs, this can help to locate objects and identify their dependency level.

If you position the mouse cursor over an object in the graph, the cursor changes to a crosshair and the object's name appears as a tool tip. Also, the connector lines radiating from the object are highlighted with yellow to indicate its parent and child dependencies (Figure 86).

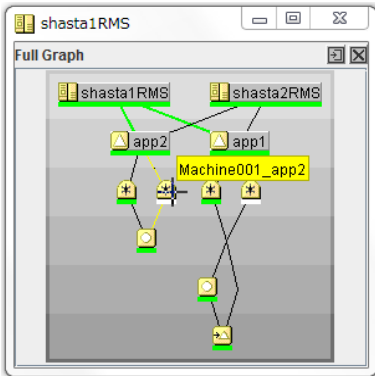


Figure 86: RMS full graph—object tooltip

Clicking on the object brings up a window with further details such as the object's attributes (Figure 87).

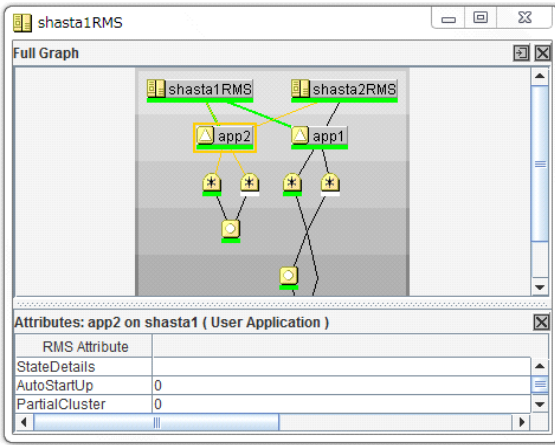


Figure 87: RMS full graph—object details

7.2.2 Application graph

- ▶ To display the graph for a single application, right-click on the application object and select *View Application Graph* from the context menu (Figure 88).

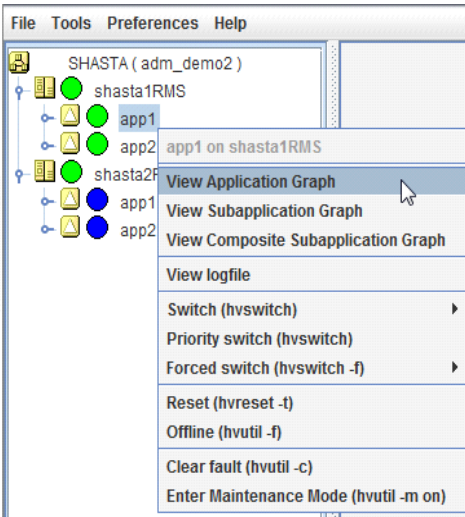


Figure 88: Viewing an RMS application graph

The application graph shares the same features as the full graph, except that it shows only the selected application and its resources (Figure 89). Like the full graph, the application graph is shown from the perspective of the selected node, and tooltips and details are available for every object.

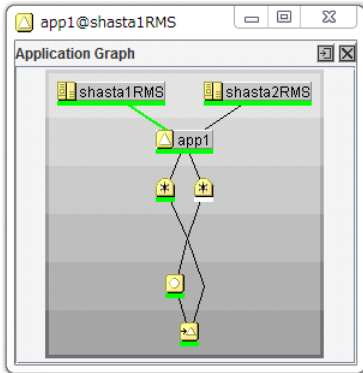


Figure 89: Typical RMS application graph

7.2.3 Subapplication graph

- ▶ To display a subapplication graph, right-click on the parent application and select *View Subapplication Graph* from the context menu (Figure 90).

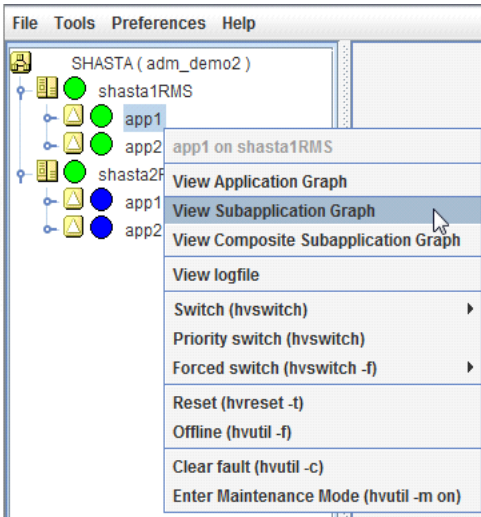


Figure 90: Viewing an RMS subapplication graph

This graph displays all the subapplications used by the selected application, showing the connections between the subapplications (Figure 91).

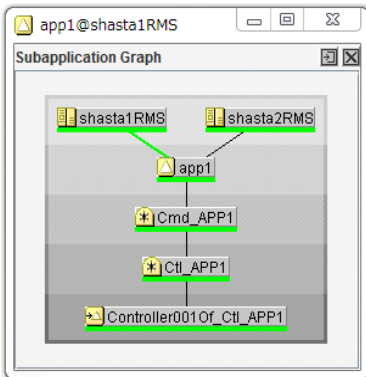


Figure 91: Typical RMS subapplication graph

For clarity, names of the subapplication objects are shown as labels rather than tooltips, and various abstractions such as non-affiliated objects are not included. Like other graphs, clicking on an object brings up a window that displays its attributes.

7.2.4 Composite subapplication graph



The composite subapplication graph is available only for applications with controller objects.

When the configuration includes a controller object, the full graph of the node (or of either application) shows the parent and child applications in separate branches (Figure 92).

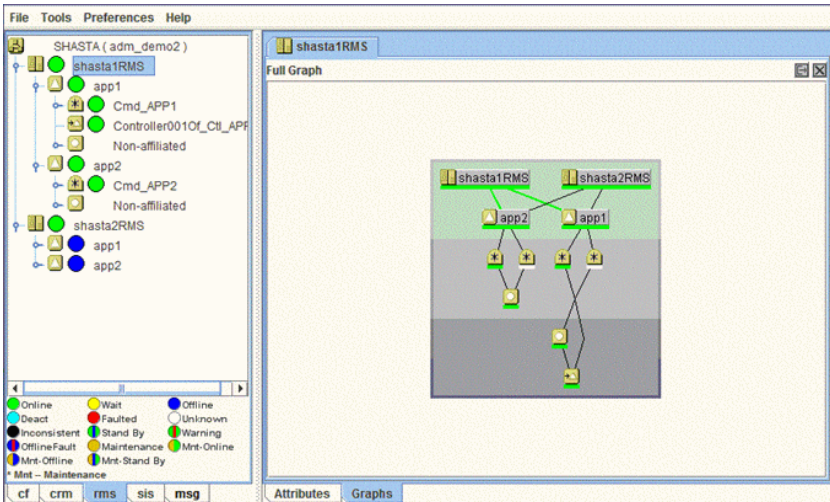


Figure 92: Standard view of configuration with controlled application

In some cases, it is convenient to view the child application as a resource of the parent so they both appear in the same branch. This is called a **composite subapplication graph**.

- ▶ To view a composite subapplication graph, right-click on the parent application and select *View Composite Subapplication Graph* from the context menu (Figure 93).

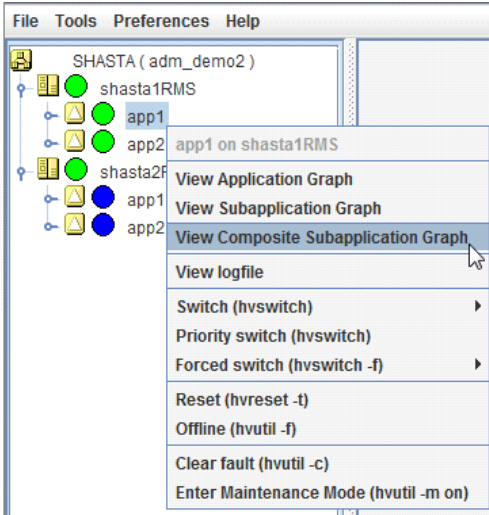


Figure 93: Viewing an RMS composite subapplication graph

For every controller object in the subapplication graph, the graph of its controlled application is inserted with a dotted line connection to the parent controller. For example, note where *app2* appears in the composite subapplication graph in Figure 94 below, and compare this to the standard graph shown earlier in Figure 92.

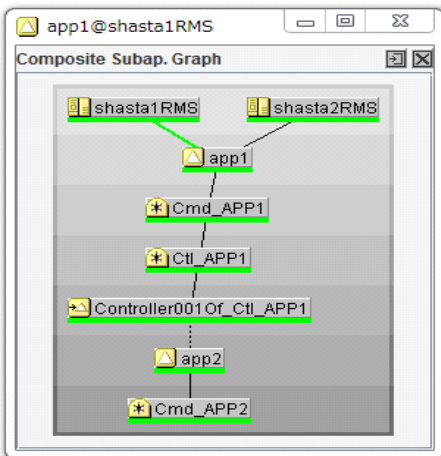


Figure 94: Typical composite subapplication graph

If the controlled application has its own controller objects, then the process is recursively repeated. This gives a composite view of all the subapplications that the selected parent application depends on, whether directly or indirectly.

7.2.5 Using pop-up context menus from the graph

Each object in the RMS graph has a pop-up context menu that provides quick access to commonly-used operations.

- To display a context menu, right-click on any object in a graph.

The first item on the menu displays the selected object's name in grayed-out text. The remaining items list the available operations, which vary according to the object's type and current state (Figure 95).

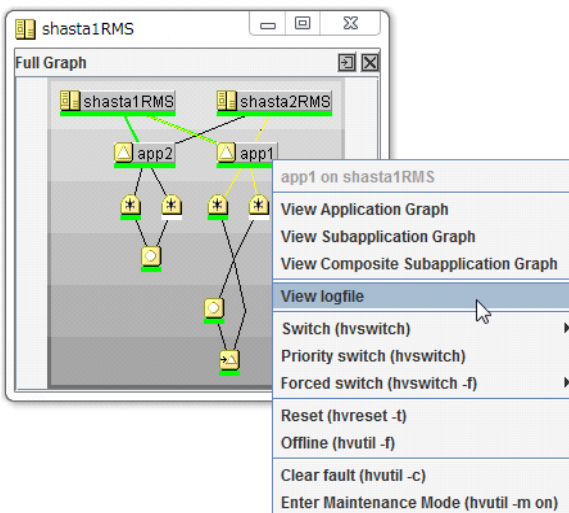


Figure 95: Using a pop-up context menu from the RMS graph

Items that affect object states, node states, or the entire RMS configuration appear toward the bottom of the menu. These actions are described in the final sections of this chapter.

- To close the context menu without performing an operation, click the grayed-out object name in the menu or press the **[Esc]** key.

7.2.6 Changing the displayed detail level

By default, the RMS graph does not display the resource (object) names on the graphs. These are available as tool tips and can be seen by placing the mouse over a particular object.

- ▶ To add any combination of resource names, affiliation names, “NoDisplay” nodes to the graphs, use the checkboxes on the *Preferences* menu.

Figure 96 shows the preference setting and a corresponding graph that displays resource names.

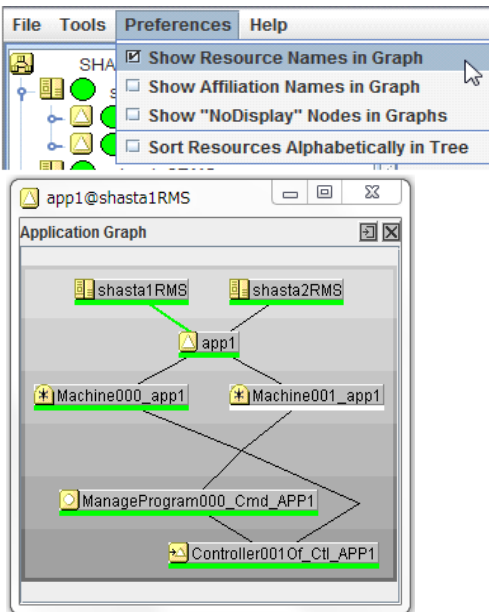


Figure 96: Displaying an RMS graph with affiliation names

Figure 97 shows the preference setting and a corresponding graph that displays affiliation names.

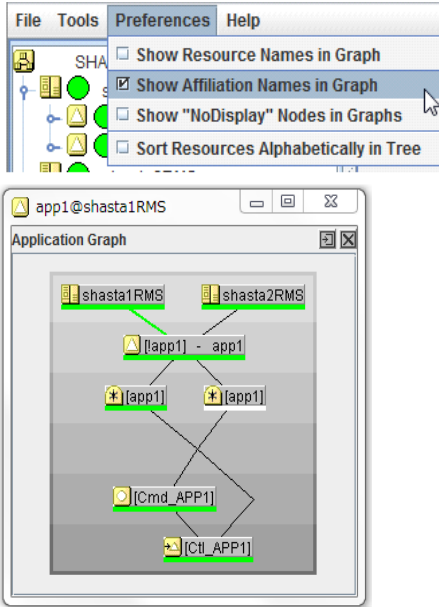


Figure 97: RMS graph with resource names

Figure 98 shows the preference setting and a corresponding graph that displays “NoDisplay” objects. These are typically logical AND/OR objects that are automatically generated by the configuration tool.

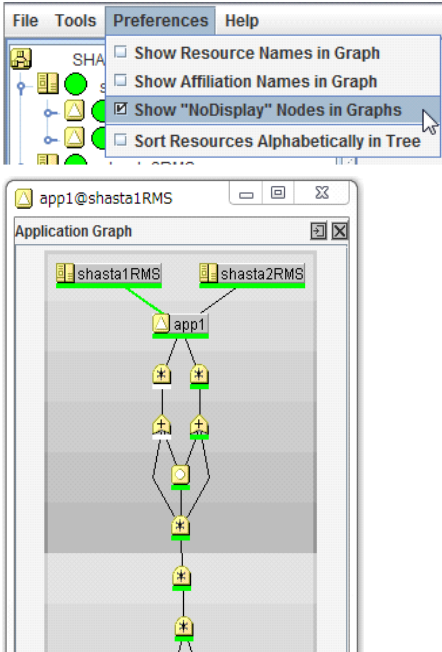


Figure 98: RMS graph with resource names

If two or more display options are selected, some of the object and resource names will consume more on-screen space (Figure 99). Horizontal scrollbars will appear when the graph is larger than the width of the screen, but the graph may still be difficult to read.

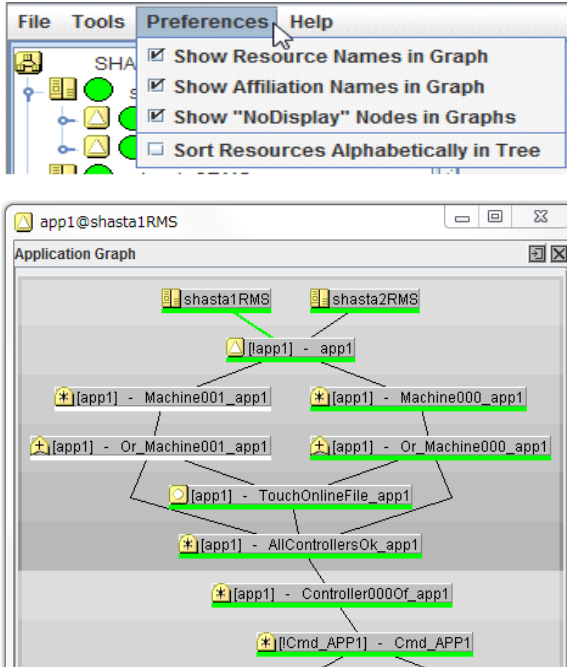


Figure 99: RMS graph with resource and affiliation names

In a complicated graph, it may help to sort the resource names alphabetically (Figure 100).

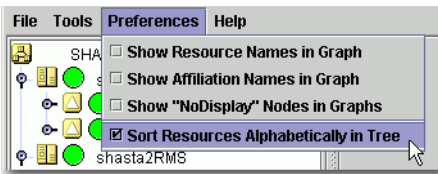


Figure 100: Sorting object names in the graph

7.3 Interpreting display changes

The Cluster Admin view, the graphs, and the clusterwide table all have methods to denote the state of individual nodes as well as the overall configuration.

7.3.1 Display during RMS configuration changes

When you stop and restart RMS with a different configuration, the RMS tree, the clusterwide table, and the node graphs are redrawn in the same windows.

Figure 101 illustrates a Cluster Admin view overlaid by two individual node graphs and the clusterwide table, all displaying the state of a running configuration that monitors app1 and app2.

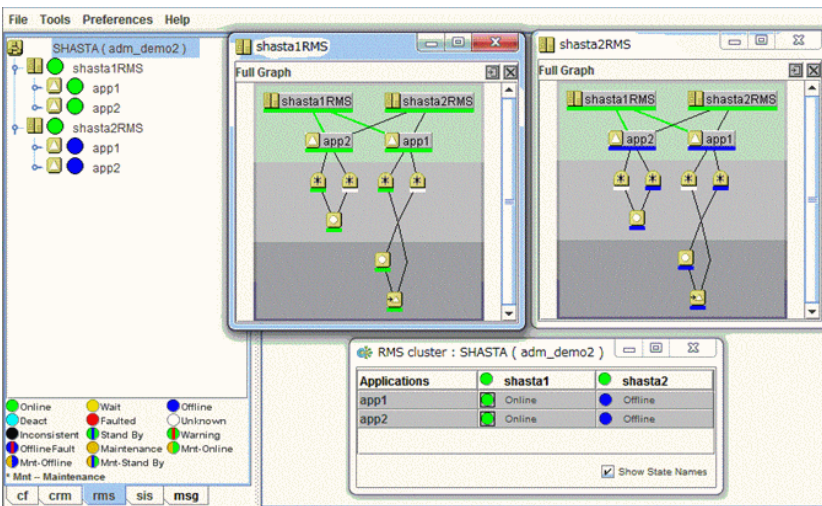


Figure 101: Cluster state before RMS is shut down

Figure 102 shows the same windows after RMS has been restarted, this time with a different configuration that monitors appA and appB.

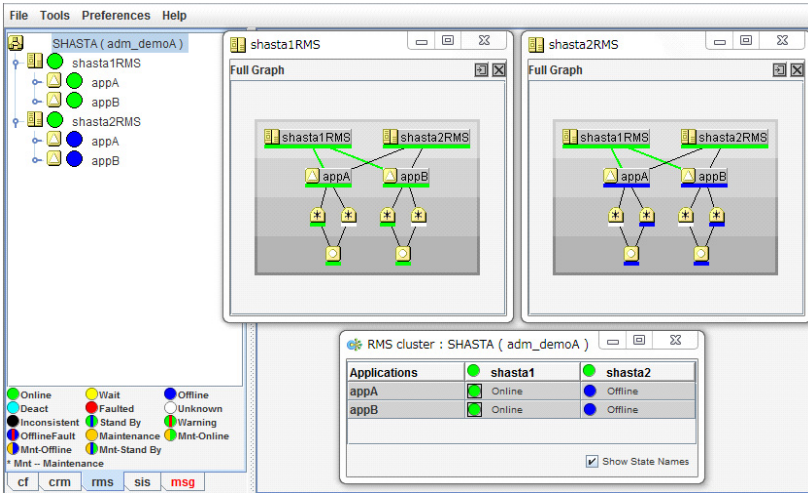


Figure 102: Cluster state after RMS restart with different configuration

The graphs and clusterwide table display the status of the same SysNode objects (shasta1 and shasta2) before and after the restart, so the windows remain open.

7.3.2 Display after RMS shutdown

After RMS is shut down, the background of RMS graph windows become dark gray on the node from which they are getting their information. In this condition, all the states are white, indicating that the states are unknown. Once the node on which RMS is activated is connected, the GUI starts to display the information again.

For example, suppose RMS is shut down only on one node (shasta1) of our example cluster. The graph on that node and the corresponding column of the clusterwide table will be shaded with a dark gray background (Figure 103).

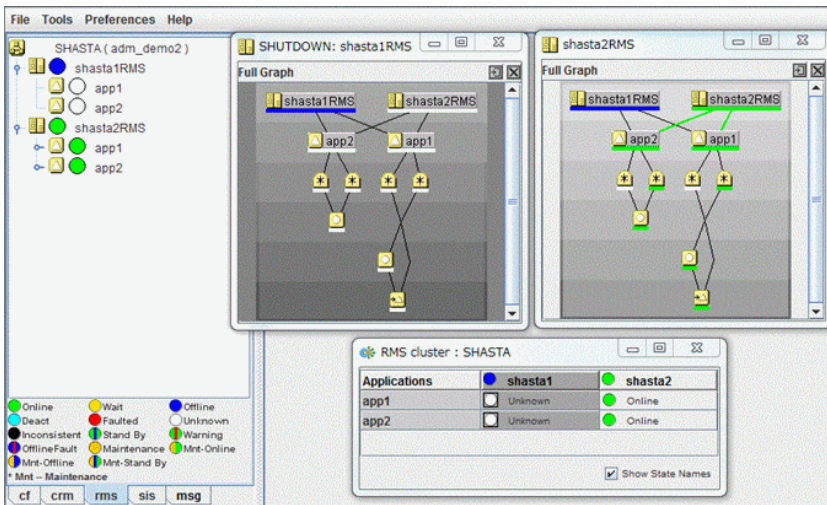


Figure 103: RMS main view, graphs, and clusterwide table after shutdown on one node

However, as long as RMS continues to run on the remaining node, *shasta2*, the RMS main view, the *shasta2RMS* graph, and the *shasta2* column in the clusterwide table will indicate the online objects.

Application and subapplication graphs

When a node shuts down, any application or subapplication graph on that node would also change to the shutdown state (Figure 104).

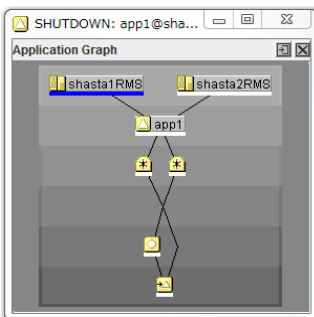


Figure 104: Application graph on shutdown node

The graph retains this appearance until the node restarts with the same configuration, at which point it returns to its normal display.

If the node restarts with a different configuration, the graph contents are deleted, because the object(s) no longer exist in the new configuration. However, the empty graph window remain opens until explicitly closed (Figure 105).

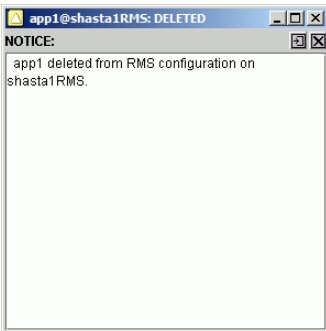


Figure 105: Graph window for deleted object

7.4 Viewing RMS log messages

The Cluster Admin interface provides a log viewer that lets you view and filter entries in the RMS switchlog and individual application logs on any node.

i All RMS log files, which normally reside in `/var/opt/SMAWRrms/1log/`, can be viewed directly using a standard UNIX editor like `vi`.

i For meanings of error messages and possible corrective actions, refer to "PRIMECLUSTER Messages."

View the switchlog for a system node as follows:

- ▶ Right-click on the system node and select *View Switchlog* from the pop-up context menu (Figure 106). Alternatively, select a node and use *Tools* → *View switchlog* (Figure 107).

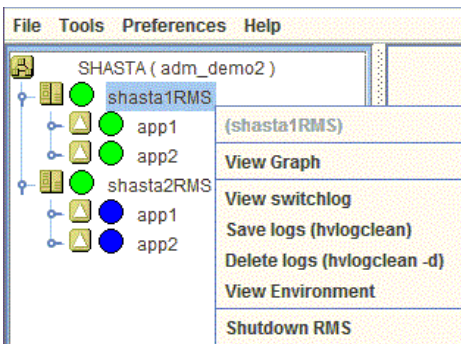


Figure 106: Viewing the RMS switchlog file using a context menu

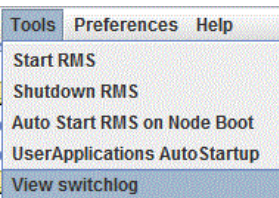


Figure 107: Viewing the RMS switchlog file using the Tools menu

View an application log as follows:

- ▶ Right-click on an application on the RMS tree and choose *View logfile* from the pop-up context menu (Figure 108).

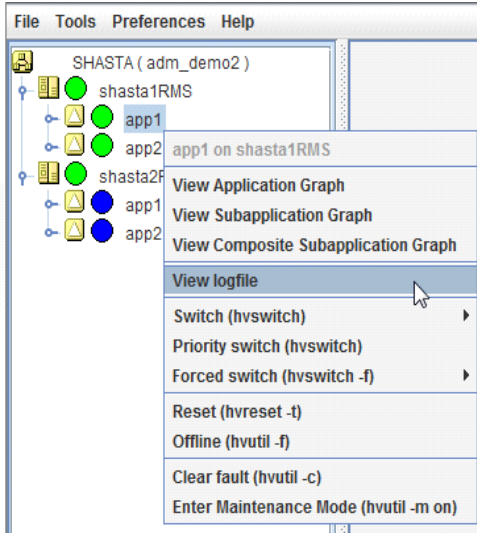


Figure 108: Viewing an application log using a context menu

i You can invoke equivalent context menus for an object from the Cluster Admin view, from the clusterwide table, or from any RMS graph containing that object.

By default, each log file is displayed in a separate tab in the right pane (Figure 109).

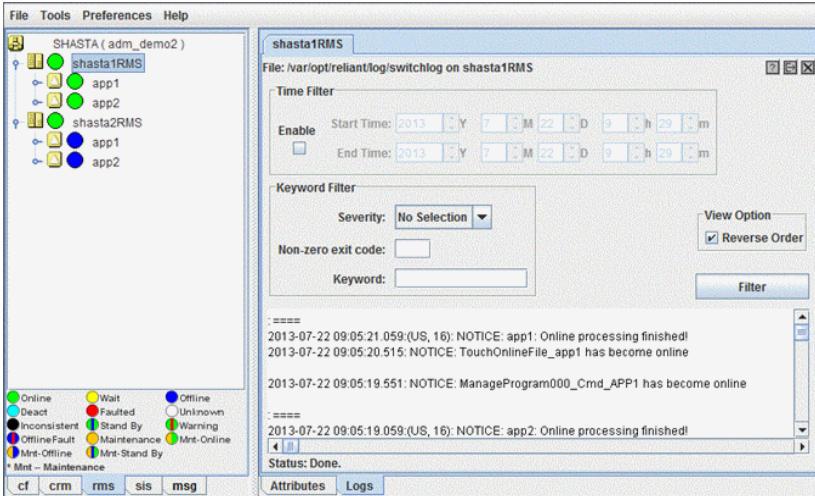


Figure 109: RMS switchlog in tab view

- ▶ To view any tab in a separate window, click the **detach** control button. The detach button is located between the **help** and **close** control buttons in the upper-right corner of the view (Figure 110).



Figure 110: Detail of tab view showing detach button

The detached view contains the same information as the tabbed view (Figure 111).

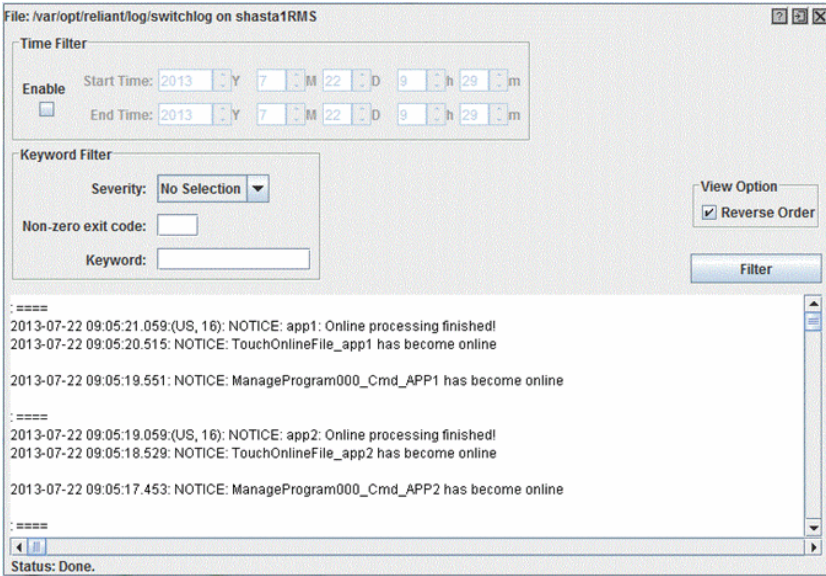


Figure 111: RMS switchlog in detached view

- ▶ To rejoin the detached window to the Cluster Admin view, click the **attach** control button. The attach button is located between the view's **help** and **close** control buttons in the upper-right corner, just below the standard window control buttons (Figure 112).



Figure 112: Detail of detached window view showing attach button

i While in detached mode, the view's close button and the standard window **close** button serve the same purpose: they both close the detached window.

In attached mode, the tabbed view's **close** button closes only the visible tab. All other tabs remain open.

7.4.1 Common procedures for switchlog and application log

By default, the entire log is available in the scrolled area at the bottom of the window. You can restrict the entries displayed with the following filters, which are described in subsections below:

- *Time Filter*—defines the time period of interest.
- *Keyword Filter*—selects a particular resource name (for an application only), error message severity level, non-zero exit code, or keyword.



For details on severity levels, see the section "Severity", and on exit codes, see "PRIMECLUSTER Messages."

- ▶ After you enter your filter criteria, click the *Filter* button to display the filtered log entries.



All the selected and non-blank *Time Filter* and *Keyword Filter* controls are combined with a logical AND operation.

At any time, you can sort the displayed switchlog entries according to increasing or decreasing time by checking or unchecking the *Reverse Order* checkbox in the log viewer window.

7.4.2 Time filter

The controls in the *Time Filter* panel allow you to limit the entries displayed in the log pane according to their date and time (Figure 113).

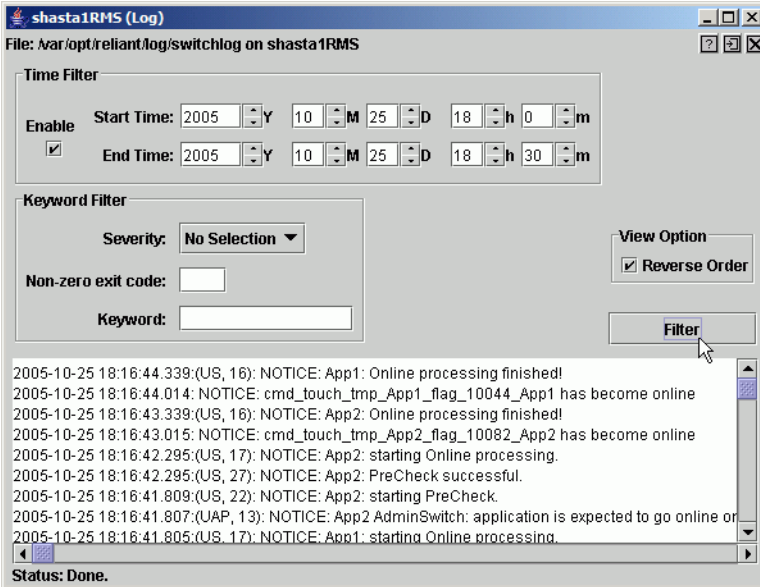


Figure 113: Search based on date and time range

- ▶ Select the *Start Time* and *End Time* using the scrolling input boxes (you can also type in the values directly) and then check the *Enable* checkbox.

The controls take effect the next time you click the *Filter* button.

- ▶ To remove the time filter, uncheck *Enable* and then click *Filter*.

7.4.3 Keyword filters

The following items are available in the *Keyword Filter* panel.

7.4.3.1 Resource Name



The *Resource Name* control is available only for application logs.

- ▶ Select a resource name from the dropdown list (Figure 114) and then click *Filter*.

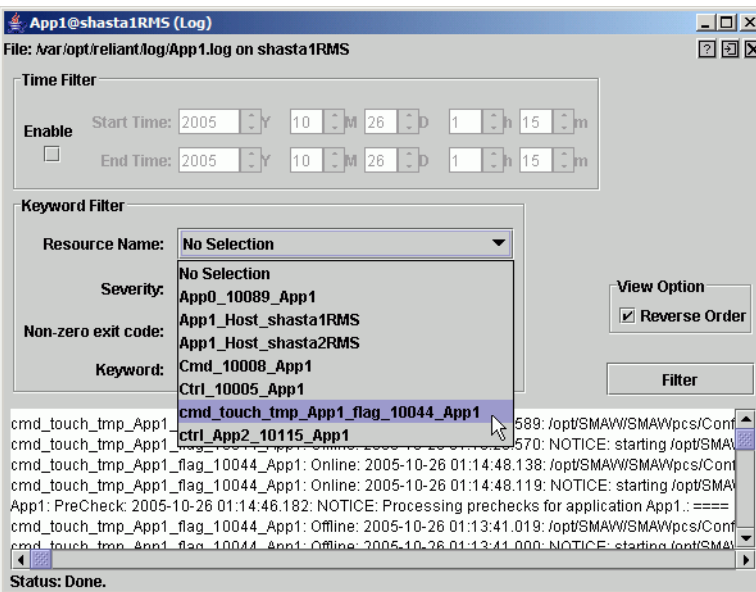


Figure 114: Search based on resource name

- ▶ To remove the resource name filter, select *No Selection* from the dropdown list and then click *Filter*.

7.4.3.2 Severity

- ▶ Select an message severity level from the dropdown list (Figure 115) and then click *Filter*.

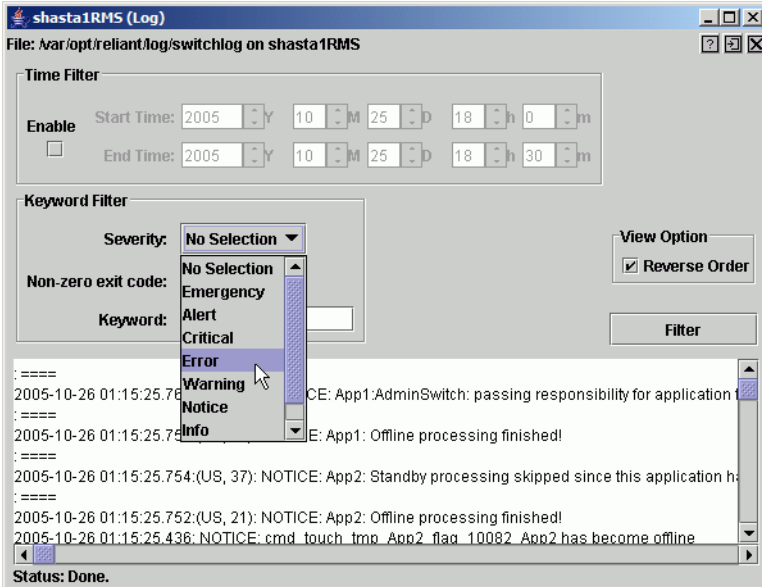


Figure 115: Search based on severity level

Table 5 summarizes the RMS message log viewer severity levels.

Severity level	Description
<i>Emergency</i>	Systems cannot be used
<i>Alert</i>	Immediate action is necessary
<i>Critical</i>	Critical condition (fatal error)
<i>Error</i>	Error condition (non-fatal error)
<i>Warning</i>	Warning condition
<i>Notice</i>	Normal but important condition
<i>Info</i>	Miscellaneous information
<i>Debug</i>	Debug messages

Table 5: RMS severity level description

- ▶ To remove the severity level filter, select *No Selection* from the dropdown list and then click *Filter*.

7.4.3.3 Non-zero exit code

- ▶ Enter a numeric exit code in the *Non-zero exit code* input box and then click *Filter*.

7.4.3.4 Keyword

- ▶ Enter a string in the *Keyword* box (Figure 116) and then click *Filter*.

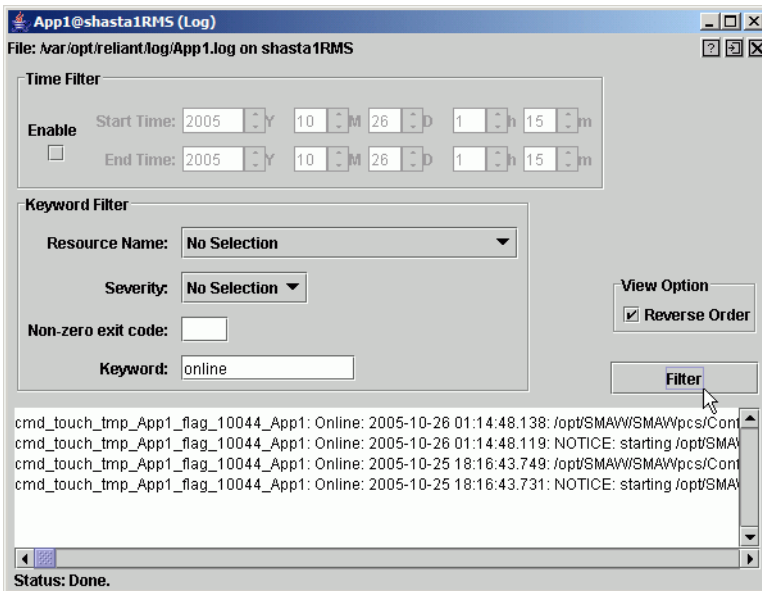


Figure 116: Search based on keyword



Special characters and spaces are valid, but wildcards are not interpreted. This search is **not case-sensitive**.

- ▶ To remove the keyword filter, clear the text in the *Keyword* box and then click *Filter*.

7.4.4 Text search

You can search the text in the application log by right-clicking on the displayed text. A pop-up dialog with a *Find* entry allows you to perform a **case-sensitive** search in the direction you specify (Figure 117).

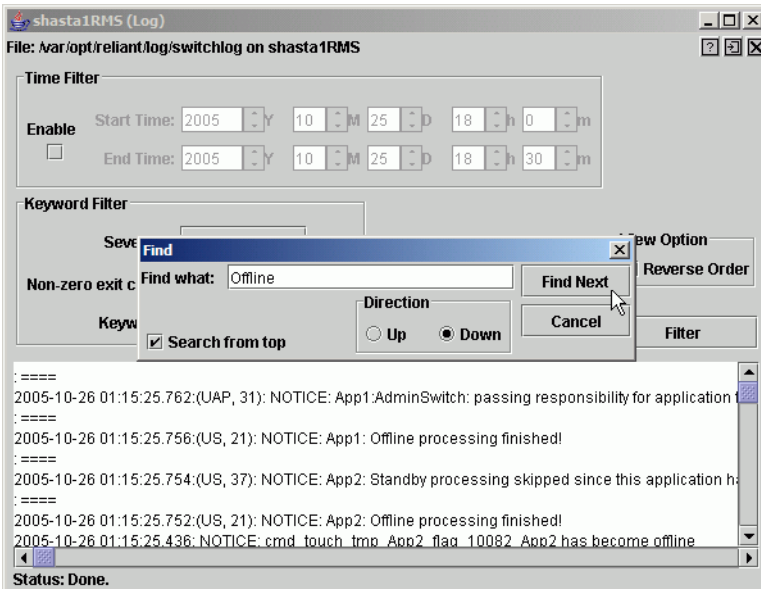


Figure 117: Using the pop-up Find dialog in log viewer



The *Find* search string is processed literally. You can include spaces and special characters, but wildcards are not interpreted.

7.4.5 Removing filters

To remove all filters, take the following steps:

- Uncheck the time filter *Enable* box.
- Set drop-down lists to *No Selection*
- Clear text from input boxes
- Click the *Filter* button

The unfiltered view will be restored.

8 Controlling RMS operation

Chapter contents:

- "Managing RMS nodes"
- "Managing RMS applications"
- "Managing resources"
- "Using maintenance mode"

8.1 Managing RMS nodes

This section describes basic procedures to control the operation of RMS, including how to start and stop individual nodes or the entire cluster. Procedures in this section are active: they change the state of the RMS cluster and may have a direct effect on the disposition of data.

As stated in the overview of this chapter, the primary means of administration is through the Cluster Admin GUI. This method should be used whenever possible. However, each procedure in this section includes a CLI alternative.

- The command is stored in `<RELIANT_PATH>/bin` directory.
- If CF code names and RMS node names follow the naming convention (if the structure of the name is `<CFname>RMS`), both are operated as Sysnode object in all of the commands of RMS CLI.
- In the description of commands, only the most frequently used options are stated. Available options may exist other than these. For command details, see the manual pages of each command.

8.1.1 Starting RMS

When you use the GUI, you can only start the most recently activated configuration. To start a different configuration, you must first use the Wizard Tools to activate that configuration.

By default, the GUI will start RMS on all the nodes in the cluster. Alternatively, you can start RMS only on a single node that you select.

- ▶ From the Cluster Admin *rms* tab, select *Tools* → *Start RMS* (Figure 118).

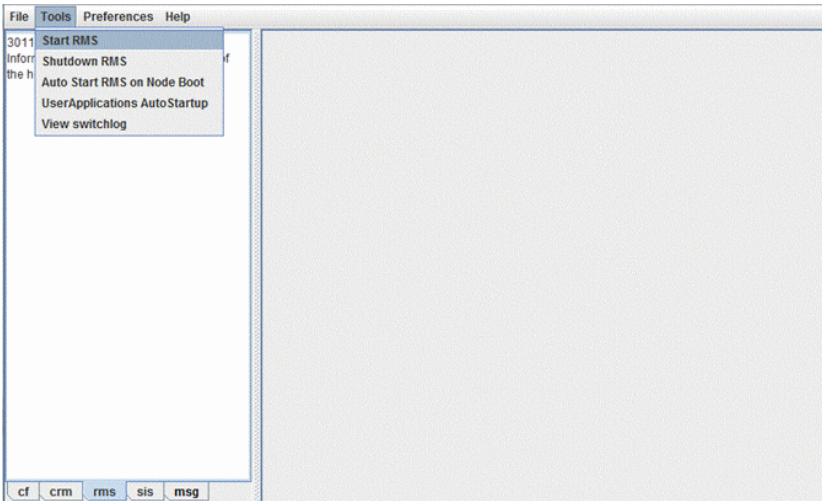


Figure 118: Starting RMS from the main menu

The *RMS Start Menu* window opens.

- ▶ To start RMS on all the nodes, click the *all available nodes* radio button and then click *OK* (Figure 119).

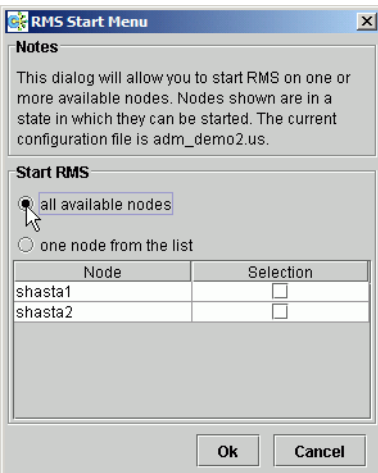


Figure 119: RMS Start Menu for all the nodes

- ▶ To start RMS only on a single node, click the *one node from the list* radio button, and then choose the node using a checkbox in the *Selection* column. After making your selections, click *OK* (Figure 120).

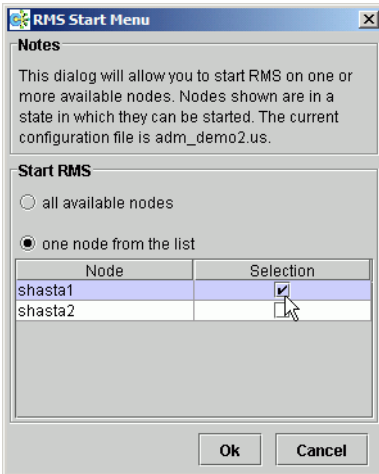


Figure 120: RMS Start Menu for individual nodes

Alternatively, you can start RMS on individual nodes directly from the *Cluster Admin* window:

1. In the left pane, click the *rms* tab to view the cluster tree.
2. Right-click on the node and select *Start RMS* from the pop-up menu (Figure 121).

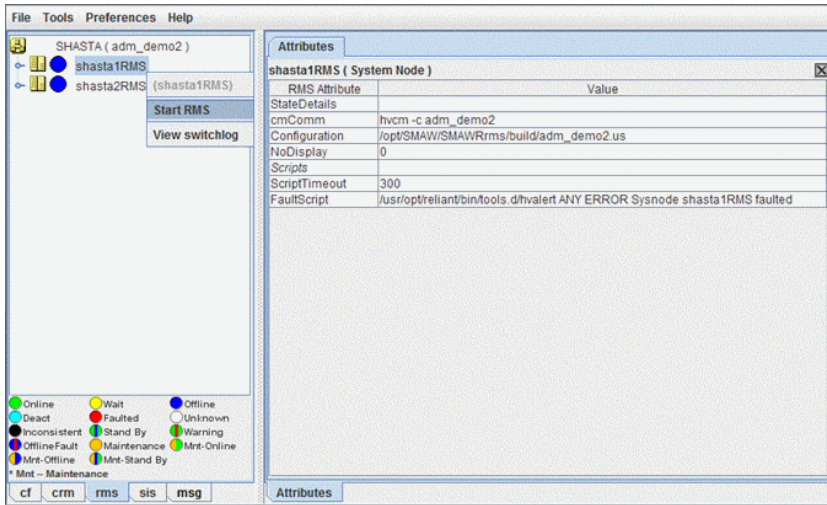


Figure 121: Starting RMS on individual nodes

CLI: hvcm

The `hvcn` command starts the base monitor and the detectors for all monitored resources. The CLI syntax for `hvcn` has two possible formats:

Format 1

```
hvcn [-a | -s SysNode]
```

Format 2

```
hvcn -c config_name [-a | -s SysNode] [-h timeout] [-l loglevels]
```

The options valid for both formats are:

- a Start RMS on all the nodes in the configuration
- s Start RMS only on the specified node

If neither '-a' nor '-s' is specified, `hvcn` starts RMS only on the local node.

The options valid only for *Format 2* are:

- c Use the specified configuration file
- h Use the specified heartbeat recovery timeout
- l Activate diagnostic output according to the specified level(s)



To start a configuration other than the one most recently activated, change the heartbeat recovery timeout, or set the diagnostic level, you must use the CLI. These parameters cannot be adjusted from the Cluster Admin GUI.

Notes for Format 1

When the `-c` option is not present, `hvc` reads the default `CONFIG.rms` startup file. `hvc` looks for the default startup file in `<RELIANT_PATH>/etc/CONFIG.rms`. If the default for the environment variable `RELIANT_PATH` has not been changed, this resolves to `/opt/SMAW/SMAWRms/etc/CONFIG.rms`. Note that the search is always confined to the local node, even if you specify the `-a` or `-s` option to start RMS remotely.

The `CONFIG.rms` file contains either of the following:

- A simple configuration name, optionally with a `.us` suffix.
- An `hvc` command compliant with *Format 2* that starts the most recently activated configuration.

You cannot specify the `-h` or `-l` options in *Format 1*. You can, however, edit the `CONFIG.rms` file and insert the options to comply with *Format 2*.

Notes for Format 2

When the `-c` option is present, and the configuration file is not an absolute path, `hvc` looks for the first match in `<RELIANT_STARTUP_PATH>`. If the default for the environment variable `RELIANT_STARTUP_PATH` has not been changed, this resolves to `/opt/SMAW/SMAWRms/build/<config_name>.us` (`hvc` adds the `.us` extension if it is not specified as part of the configuration file name). If an absolute path is specified, `hvc` attempts to read only that file. Note that the search is always confined to the local node, even if you specify the `-a` or `-s` option to start RMS remotely.

When `hvc` locates the specified configuration file, it checks to see if the default `CONFIG.rms` file is also present. If it is, `hvc` compares the configuration names defined by both files to make sure they agree. If they do not agree, `hvc` aborts the startup processing.

The `-h` option sets the UDP heartbeat recovery timeout for the cluster. See "Nodes and heartbeats". The default timeout depends on which method RMS uses to monitor the cluster:

- By default, RMS monitors the state of each node with the Enhanced Lock Manager (ELM), which is implemented in the Cluster Foundation. ELM is not a polling method. Instead, it relies on locks that are held by each node and released when the node or its base monitor go down. When ELM is enabled, the UDP heartbeat timeout defaults to 600 seconds.
- ELM can be disabled for rolling upgrade or debugging operations by setting the `HV_USE_ELM` environment variable to zero (0). See "`HV_USE_ELM`". When ELM is disabled, the UDP heartbeat defaults to 45 seconds.

i Specifying a heartbeat timeout shorter than the default may cause premature node kills. No data loss will occur, because a node kill begins with a graceful shutdown. However, cluster performance may suffer due to latency as applications are switched to different nodes. Excessively short heartbeat timeouts may also interfere with CF event timeouts.

The `-1` option sets the diagnostic output level at startup time. The *loglevels* specification consists of one or more individual numeric levels or hyphen-delimited ranges, each separated by a comma.

8.1.2 Starting RMS automatically at boot time



This setting takes effect at the next system startup

You can use the following procedure to activate or deactivate automatic RMS startup when the system boots up.

- ▶ From the Cluster Admin *rms* tabbed view, select *Tools > Auto Start RMS on Node Boot* (Figure 122).

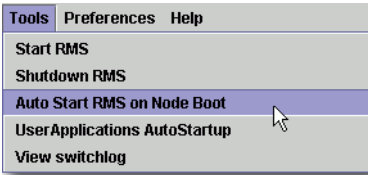


Figure 122: Controlling automatic RMS startup—step 1

You can then choose to activate (or deactivate) the automatic RMS startup on all the nodes, or on just one node (Figure 123).

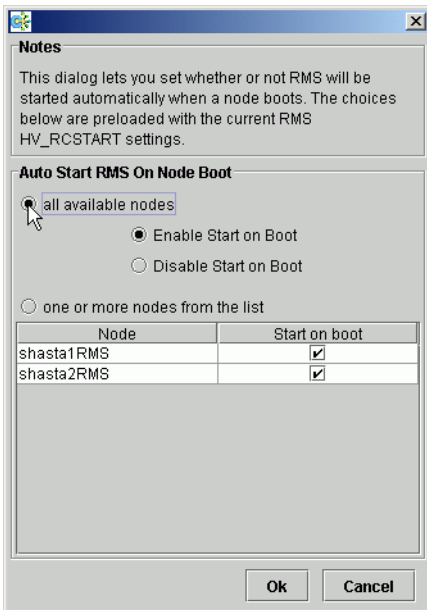


Figure 123: Controlling automatic RMS startup—step 2



To enable this setting, Web-Based Admin View and CF on all the nodes that configure the cluster need to be working.

For how to check the server process status of Web-Based Admin View, and how to start Web-Based Admin View, see "Initial Setup of Web-Based Admin View" in "PRIMECLUSTER Installation and Administration Guide."

For how to check the status of CF and how to start CF, see "Main CF table" and "Starting and stopping CF" in "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide."

CLI: `hvsetenv`

At system startup, the RMS `rc` script checks the environment variable settings: if the `HV_RCSTART` environment variable is set to 1, the `rc` script will attempt to start RMS using the `CONFIG.rms` file. You can set the `HV_RCSTART` variable with the `hvsetenv` command as follows:

```
hvsetenv HV_RCSTART [0|1]
```

The allowable values are:

- 0 Do not start RMS at boot time
- 1 Start RMS at boot time (default)

If no value is specified, the command reports the current value of the `HV_RCSTART` environment variable.

8.1.3 Stopping RMS

You can stop RMS on all the nodes or on a subset that you select.

Use the *Tools* pull-down menu (Figure 124) and select *Shutdown RMS*.

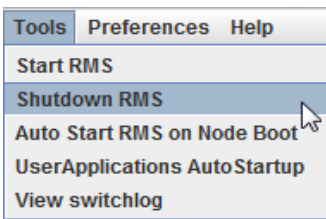


Figure 124: Using the Tools menu to stop RMS

- ▶ To stop RMS on all the nodes, click the radio button for *all available nodes* and then click *Ok* (Figure 125).

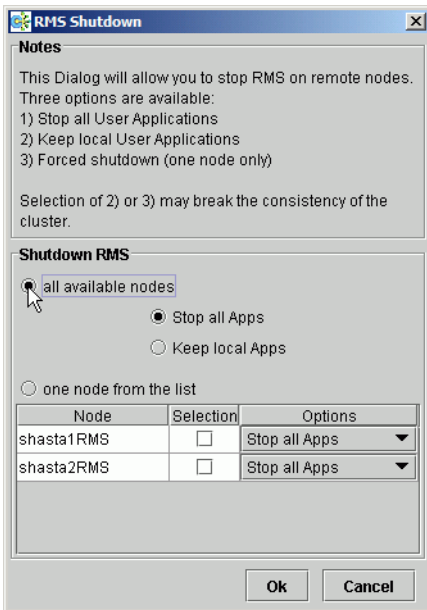


Figure 125: Stopping RMS on all available nodes

When you shut down *all available nodes*, two radio buttons allow you to choose how you want to handle the applications:

- *Stop all Apps*—Stops all user applications
- *Keep local Apps*—Leaves the applications running



Caution

Leaving the applications running after stopping RMS can lead to data inconsistencies or corruption.

To stop RMS on one specific node, select the radio button for *one node from the list*, and then click the checkbox of the node you want to shut down (Figure 126).

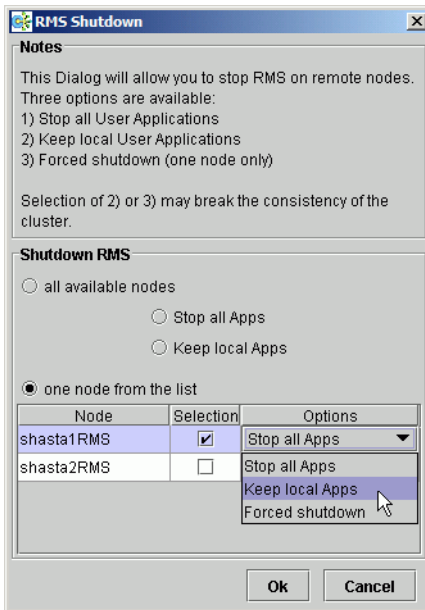


Figure 126: Stopping RMS on one node from the list

Each node has a dropdown list in the *Options* column to provide additional control:

- *Stop all Apps*—Stops all user applications on the selected node
- *Keep local Apps*—Leaves the applications running on the selected node
- *Forced shutdown*—Performs a forced shutdown of RMS



Caution

- *Keep local Apps* or *Forced shutdown* can cause data inconsistencies or corruption.
 - Do not stop RMS while RMS is running. Heartbeats between nodes are interrupted and the node where RMS is stopped may be forcibly shut down.
- ▶ Click the *Ok* button to initiate the shutdown with your selections.

Stop all Apps is the default option for shutting down RMS on all the nodes or on one node. If you select an option other than the default, you will be prompted to confirm the operation (Figure 127 and Figure 128).

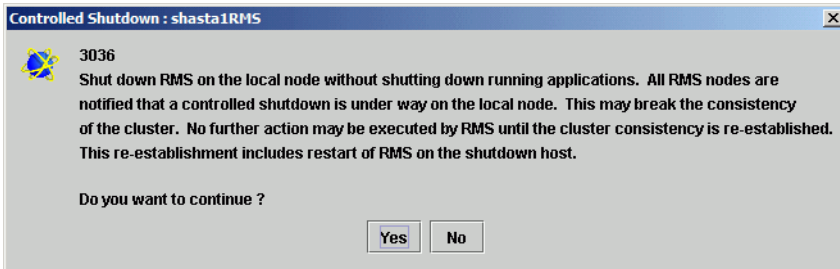


Figure 127: Stopping RMS while keeping applications—confirmation

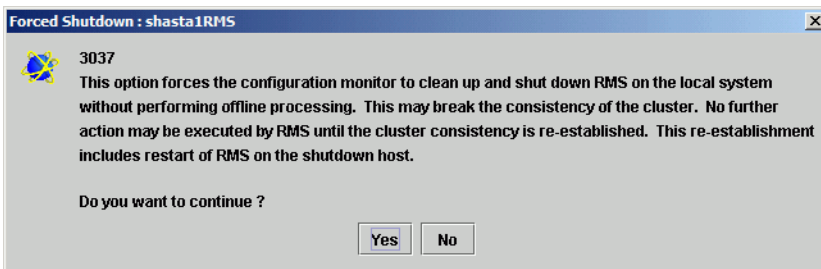


Figure 128: Forced shutdown of RMS—confirmation

You can also stop RMS on a single node by right-clicking on the node in the RMS tree and then selecting *Shutdown RMS* from the context menu (Figure 129).

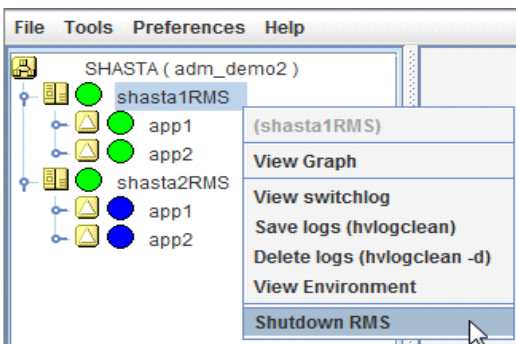


Figure 129: Using the context menu to stop RMS on one node

Only one node will appear in the confirmation window (Figure 130).

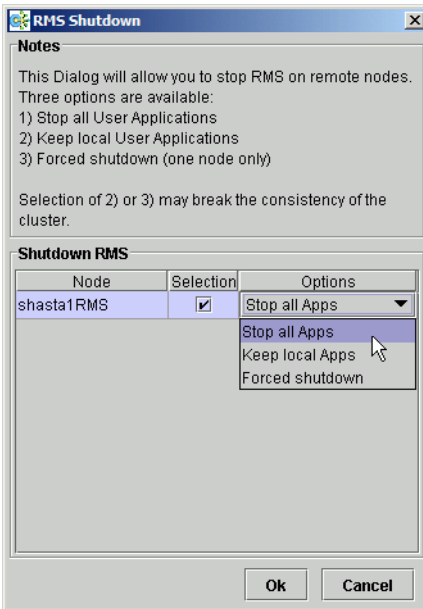


Figure 130: Stopping RMS on one node



Note

When switching online a controlling application with a scalable controller, do not stop RMS during the startup processing. When stopping RMS, the stop processing may timeout.

CLI: hvshut

The syntax for the CLI is as follows:

```
hvshut {-a | -A | -f | -l | -L | -s SysNode}
```

Options:

- a Shut down RMS and applications on all the nodes
- A Shut down RMS on all the nodes without shutting down applications
- f Forced (emergency) shutdown of RMS on the local node
- l Shut down RMS and applications on the local node
- L Shut down RMS on the local node without shutting down applications
- s Shut down RMS only on the specified node

The `hvshut` command shuts down the RMS software on one or more nodes in the configuration. The base monitor on the local node sends a message to other online nodes indicating which node or nodes are to be shut down. The `hvshut` command disables all error detection and recovery routines on the nodes being shut down, but does not shut down the operating system.

If any `userApplication` objects are online when the `-A`, `-f`, or `-L` options are used, the applications remain running but are no longer monitored by RMS. Both The `-f` and `-L` options affect only the local node, but the `-f` option is for emergencies (when other `hvshut` options do not work).

When you choose to shut down RMS without shutting down the monitored applications, you will be prompted to confirm the operation.

**Caution**

Use the `hvshut -A`, `-f`, and `-L` options carefully as they could result in inconsistencies or data corruption.

**Note**

When you stop RMS in a configuration where the operating `userApplication` exists on 3 or more nodes, perform one of the following operations.

- To stop RMS of all nodes
Perform `hvshut -a` command with one of the nodes.
- To stop RMS of 2 or more nodes (not of all nodes)
Perform `hvshut` command on each node to stop RMS. After `hvshut` command returns on a certain node, perform `hvshut` command on the next node.

8.1.4 Clearing a SysNode Wait state

You can clear a `Wait` state of `SysNode` by the following procedure:

1. Manually stop the node in the `Wait` state. Make sure the node has stopped completely.

2. Check the CF state in the CF main window.
If the CF state is `LEFTCLUSTER`, clear `LEFTCLUSTER` in the CF main window and make sure the node state is changed from `LEFTCLUSTER` to `DOWN`.
For more information on `LEFTCLUSTER` state, refer to "PRIMECLUSTER Cluster Foundation Configuration and Administration Guide."
3. If the `Wait` state of `SysNode` is not cleared after you perform step 2, right-click on the `SysNode` displayed on the RMS main window and select [Clear Wait & shutdown(`hvutil -u`)] from the pop-up context menu, and then select the `Wait` state node from the sub menu.

**Caution**

When you manually clear a `Wait` state in step 3, RMS and CF consider that the target node has been stopped. Therefore, if you clear the `Wait` state while the target node is not actually stopped, data corruption may occur.

CLI: `hvutil -u`

The syntax for performing the operation in step 3 with CLI is as follows:

```
hvutil -u SysNode
```

8.2 Managing RMS applications

This section describes basic procedures related to starting, stopping, and clearing special states of individual applications. Procedures in this section are active: they change the state of the RMS cluster and may have a direct effect on the disposition of data.

As stated in "Overview", the primary means of administration is through the Cluster Admin GUI. This method should be used whenever possible. However, each procedure in this section includes a CLI alternative.


8.2.1 Overriding automatic application startup

The automatic startup of each application is controlled by its `AutoStartUp` attribute:

- If `AutoStartUp` is set to "yes," the application starts automatically when RMS starts.
- If `AutoStartUp` is set to "no," the application does not start automatically when RMS starts (must be started manually).

For maintenance or troubleshooting, you may want to suppress the automatic application startup for all applications during RMS startup. In such cases, you can disable the `AutoStartUp` attribute of all applications at once by changing the value of the RMS environment variable `HV_AUTOSTARTUP`.

The `HV_AUTOSTARTUP` environment variable can be changed with the following procedure.

 The following procedure changes the `HV_AUTOSTARTUP` global environment variable. Changes to `HV_AUTOSTARTUP` do not take effect until the next RMS startup.

- ▶ From the Cluster Admin *rms* tab, select *Tools* → *UserApplications AutoStartup* (Figure 131).

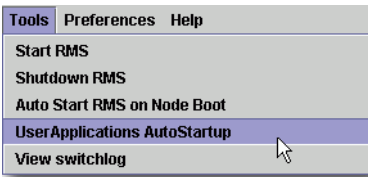


Figure 131: Controlling automatic application startup—step 1

You can then choose to override all `AutoStartUp` settings, or to cancel the override (Figure 132).

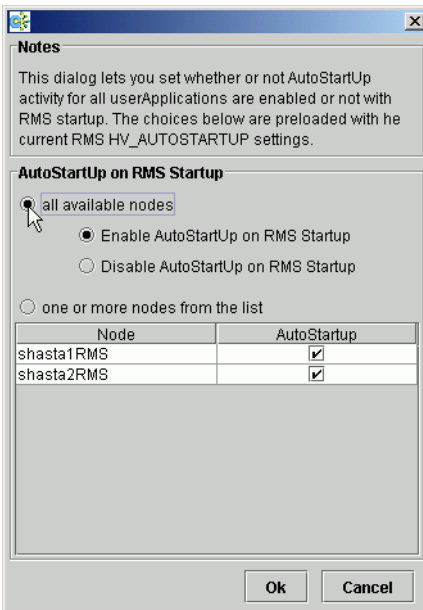


Figure 132: Controlling automatic application startup—step 2



Note

- Even if the userApplication's autostart is enabled, the userApplication where `AutoStartUp` is set to "no" needs to be started manually.
- The setting for autostart of userApplication is not intended to set the start condition of Standby processing when the RMS starts. For the start condition of Standby processing when RMS starts, follow

`AutoStartUP` and `StandbyTransition` attributes. For details, see "6.10.1 Setting Contents of a Cluster Application" in "How to start up a userApplication automatically in the event of a startup, switchover, and clearing a fault" of "PRIMECLUSTER Installation and Administration Guide."



To enable this setting, Web-Based Admin View and CF on all the nodes that configure the cluster need to be working.

For how to check the server process status of Web-Based Admin View, and how to start Web-Based Admin View, see "4.3.3 Initial Setup of Web-Based Admin View" in "PRIMECLUSTER Installation and Administration Guide."

For how to check the status of CF and how to start CF, see "4.2 Main CF table" and "4.6 Starting and stopping CF" in "PRIMECLUSTER Cluster Foundation (CF) Configuration and Administration Guide."

CLI: `hvsetenv`

The action of each application's `AutoStartUP` attribute is controlled by the `HV_AUTOSTARTUP` environment variable (see the description in "Local environment variables"). You can set this variable with the `hvsetenv` command as follows:

```
hvsetenv HV_AUTOSTARTUP [0|1]
```

The allowable values are:

- 0 Prevents automatic application startup at next RMS startup
- 1 Conducts autostart of application in accordance with `AutoStartUP` attribute on the next time RMS starts

If no value is specified, the command reports the current value of the `HV_AUTOSTARTUP` environment variable.

8.2.2 Switching an application

When you switch an application, RMS performs the following tasks:

- If the application is already running in the cluster, RMS shuts it down.
- After the application is completely shut down, RMS starts it on the node you specified.

Switch an application to any node as follows:

- ▶ Right-click on the application object on any node and select *Switch* from the context menu. A secondary menu appears, listing the available target nodes for switchover.
- ▶ Select the target from the secondary menu to switch the application to that node (Figure 133).

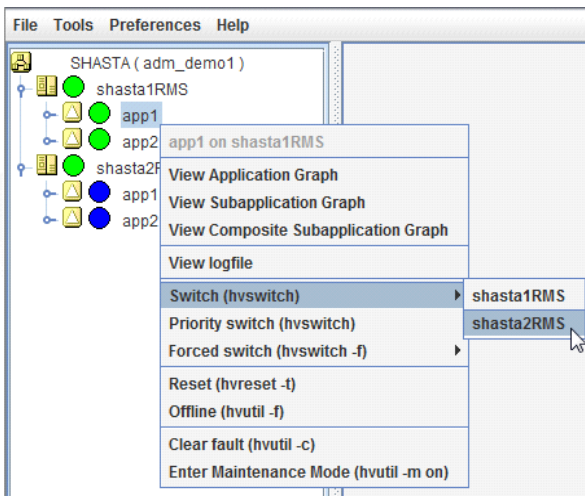


Figure 133: Switching an application

You will be prompted to confirm the action before RMS begins the operation.

A *Priority switch* switches the application to the target node according to the order of nodes in the application's `PriorityList` attribute.

If stopping the application fails, RMS does not start the application. This is because two competing instances of the same application could cause data corruption. In this event, you can start the application by using the *Forced switch* operation.



Caution

Use the *Forced switch* mode only if an application cannot be switched normally. A forced application switch overrides all safety checks and could therefore result in data corruption or other inconsistencies.

Therefore, RMS may kill the node on which RMS is not running before starting the application to reduce the risk of data corruption when the *Forced switch* request of an application is issued.

If the application is busy, the pop-up context menu will not offer the choices to switch the application. Instead, the menu will offer view-only operations, and the last menu item will indicate that the application is in a `Wait` state (Figure 134).

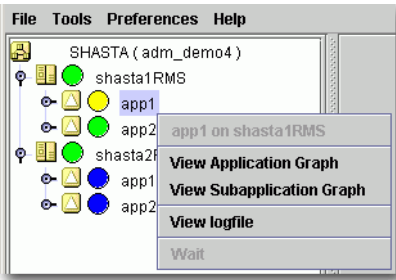


Figure 134: Switching a busy application

CLI: hvswitch userApplication

The syntax for the CLI is as follows:

```
hvswitch [-f] userApplication [SysNode]
```

The `hvswitch` command manually switches control of a `userApplication` or a `gResource` between `SysNodes` in the RMS configuration. When a type of the resource to be switched is `userApplication`, if no `SysNode` is specified, the application is switched to the highest priority node. The `-f` option is a forced-switch option.



Caution

Use the `‘hvswitch -f’` operation carefully. A forced switch of an application or a resource overrides all safety checks and could therefore result in data corruption or other inconsistencies. Therefore, RMS may kill the node on which RMS is not running before starting the application to reduce the risk of data corruption when the *Forced switch* request of an application or a resource is issued.

8.2.3 Starting an application

If the application is already offline everywhere in the cluster, you can start it (bring it online) on a single node as follows:

- ▶ Right-click on the application object and select *Online* from the pop-up context menu (Figure 135).

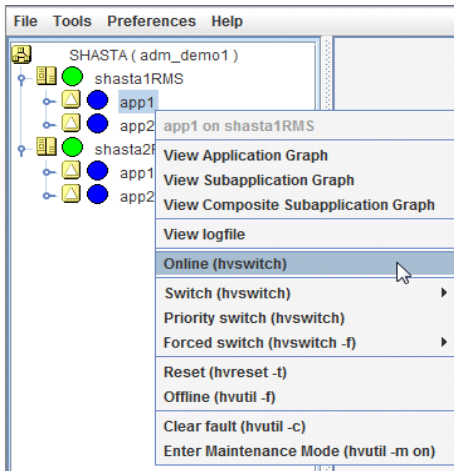


Figure 135: Starting an application

You will be prompted to confirm the action before RMS begins the operation.

CLI

Starting an application on the local node, like switching an application to another node, use the `hvswitch` command. Refer to "CLI: `hvswitch userApplication`" for the syntax.

8.2.4 Stopping an application

Stop an online application (take it offline) as follows:

- ▶ Right-click on the online application object and select *Offline* from the pop-up context menu (Figure 136).

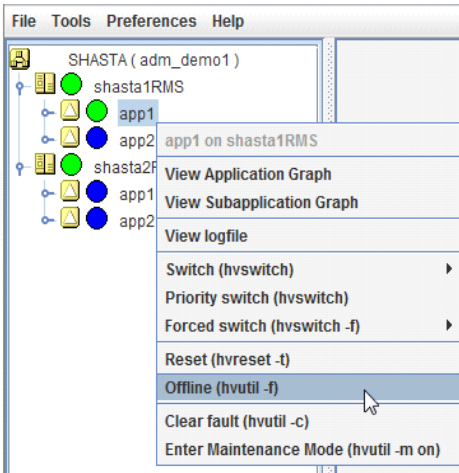


Figure 136: Shutting down an application

You will be prompted to confirm the action before RMS begins the operation.

CLI: hvutil -f

The syntax for the CLI is as follows:

```
hvutil -f userApplication
```

Note that this is a **normal offline request**. There is no “forced” offline request for a userApplication.



Use the command ‘`hvutil -s userApplication`’ to bring an offline userApplication to a Standby state.

8.2.5 Resetting an application

This operation completely re-initializes the target application and all objects in its resource tree (*i.e.*, the entire application graph) based on the actual detector reports. It will interrupt any ongoing RMS processing for the application, and any running scripts will be terminated.



Caution

Resetting an application will cause information about previous failures or any other history to be lost. It will most likely result in an *Inconsistent* state for the application.

This operation is intended for use by an experienced administrator during a test phase. It should never be invoked in a production environment.

Reset (reinitialize) an online application as follows:

- ▶ Right-click on the online application object and select *Reset* from the pop-up context menu (Figure 137).

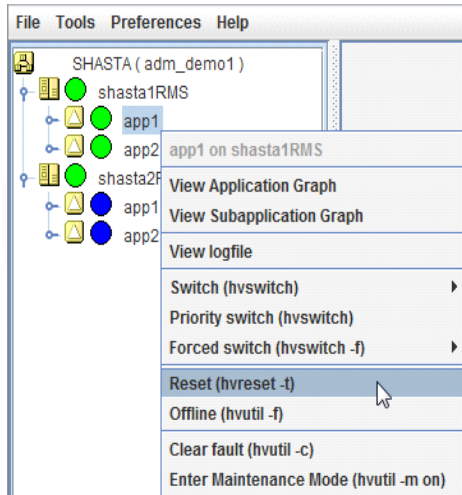


Figure 137: Resetting an application

You will be prompted to enter the reset timeout before RMS begins the operation (Figure 138).

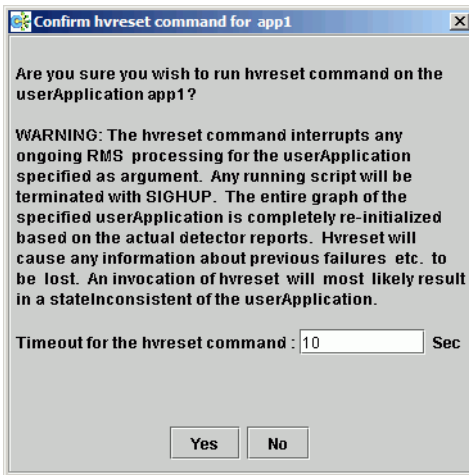


Figure 138: Choosing the reset timeout for the application

If RMS fails to re-initialize the entire application graph during this amount of time, it terminates with an error message. Default: 10 seconds

- ▶ Adjust the timeout value if desired, and then click *Yes* to reset the application.

CLI: `hvreset -t`

The syntax for the CLI is as follows:

```
hvreset -t timeout userApplication
```

`hvreset` displays a message that warns of the consequences and then prompts you to confirm the action before proceeding.

When the `userApplication` is not performing online, offline, or fault processing, `hvreset` returns immediately and displays a message stating there is nothing to do. This case is considered to be successful command execution and returns exit code 0.

8.2.6 Clearing a fault

For an application that is in the `Faulted` state, clear the fault as follows:

- ▶ Right-click on the application object and select *Clear Fault* from the pop-up context menu (Figure 139).

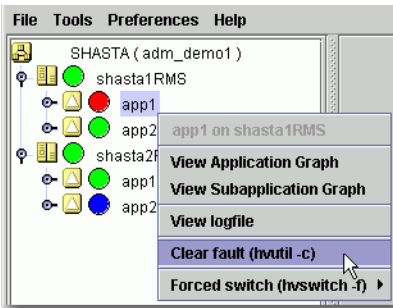


Figure 139: Clearing an application fault

Before the command proceeds, you will be informed of the action to be taken and prompted to confirm the operation (Figure 140).

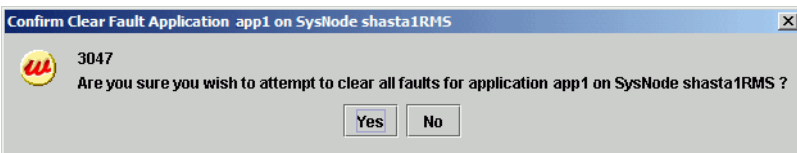


Figure 140: Clearing an application fault—confirmation dialog

CLI: hvutil -c

The syntax for the CLI is as follows:

```
hvutil -c userApplication
```

i If the *userApplication* is in the online state, then clearing the fault will cause RMS to attempt to bring the faulted resource to the online state. If the *userApplication* is in the offline or faulted state, then clearing the fault will attempt to bring the resource to the offline state. You will be informed of the action that is to be taken and prompted to confirm the operation before the command proceeds.

8.2.7 Activating an application

Activating an application takes it from the `Deact` state to the `Offline` state. It does not bring it `Online`. Also, activating a `userApplication` with the Cluster Admin GUI or the CLI has nothing to do with activating an RMS configuration in the Wizard Tools—the two operations are completely independent. Activate a deactivated application as follows:

- ▶ Right-click on the application object and select the *Activate* option from the pop-up menu.

CLI: `hvutil -a`

The syntax for the CLI is as follows:

```
hvutil -a userApplication
```



You will not need to activate an application unless someone explicitly deactivated it with the command `'hvutil -d userApplication'`.

8.3 Managing resources

This section describes basic procedures for starting and stopping of each resource. Procedures in this section are active: they change the state of the RMS cluster and may have a direct effect on the disposition of data.

As stated in "Overview" , the primary means of administration is through the Cluster Admin GUI. This method should be used whenever possible. However, each procedure in this section includes a CLI alternative.



Note

When resources under the `userApplication` are partially `Online` , the `userApplication` becomes the `Online` state.

8.3.1 Switching a resource

When you switch a resource, RMS performs the following tasks:

- If the `userApplication` that controls the resource is already started, RMS will stop it.
- After the `userApplication` is completely stopped, RMS will start the resource on the specified node.

Switch the resource in the `Online` state according to the following procedure:

Right-click on the `gResource` object and select *Online resource* from the context menu (Figure 141).

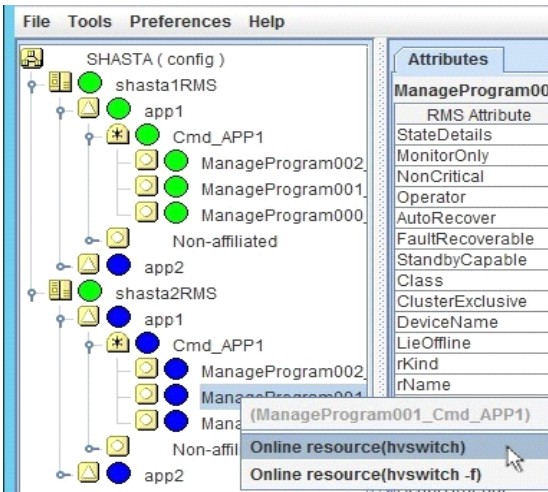


Figure 141: Switching a resource

You will be prompted to confirm the action before RMS begins the operation (Figure 142). Starts all the resources which the specified resource depends on, and then starts the specified resource.

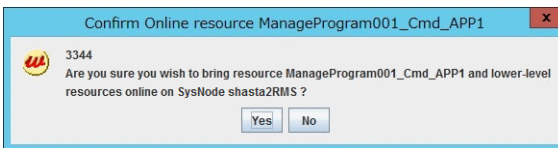


Figure 142: Confirming a resource switchover

If stopping the `userApplication` fails, RMS does not start the resource. This is because two competing instances of the same application could cause data corruption. In this event, you can start the resource by selecting *Forced Resource Online*.



Note

Use the *Forced switch* mode only if a resource cannot be switched normally. A forced resource switch overrides all safety checks and could therefore result in data corruption or other inconsistencies.

Therefore, to reduce the risk of data corruption when the Forced switch request of a resource is issued, the resource may be forcibly started after the node on which RMS is not running is forcibly stopped.

CLI: hvswitch resource

The syntax for the CLI is as follows:

```
hvswitch [-f] resource SysNode
```

The `hvswitch` command manually switches control of a `userApplication` or a `gResource` between `SysNodes` in the RMS configuration. When a type of the resource to be switched is `gResource`, the `SysNode` name of the node on which this command is executed should be specified. The `-f` option is a forced switch option.



Note

The resource can be switched only when the application to control the resource is `Online`, `Offline`, or `Standby`.

8.3.2 Starting a resource

If a resource is in the `Offline` state or the `Standby` state, you can start it (bring it to the `Online` state) on a single node as follows.

Right-click on the `gResource` object and select *Online resource* from the pop-up context menu (Figure 143).

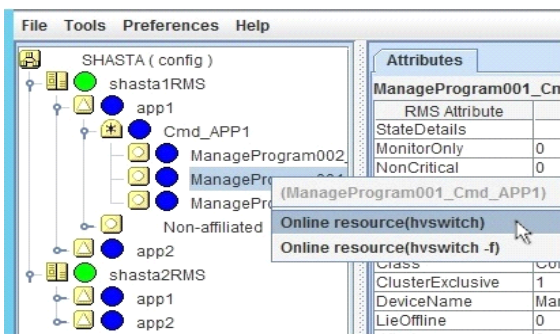


Figure 143: Starting a resource

You will be prompted to confirm the action before RMS begins the operation (Figure 144). Starts all the resources which the specified resource depends on, and then starts the specified resource.

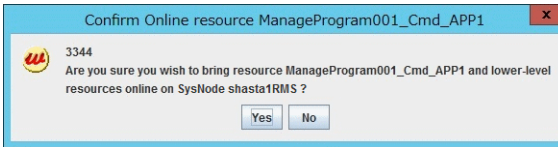


Figure 144: Confirming a resource start

CLI

You can use the `hvswitch` command to start a resource on the local node, like switching a resource to another node. See "CLI: hvswitch resource".



Note

- The resource can be started only when the application to control the resource is `Online`, `Offline`, or `Standby`.
- If some resource becomes `Online` unexpectedly while the `userApplication` is partially `Online`, you cannot specify other inactive resource to start the resource.

8.3.3 Stopping a resource

Stop the online resource (change the resource to offline) as follows:

- ▶ Right-click on the online `gResource` object and select *Offline Resource* from the pop-up context menu (Figure 145).

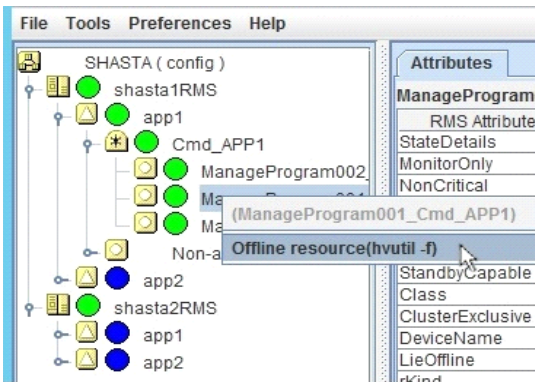


Figure 145: Stopping a resource

You will be prompted to confirm the action before RMS begins the operation (Figure 146). Stop all the resources which the specified resource depends on, and then stop the specified resource.

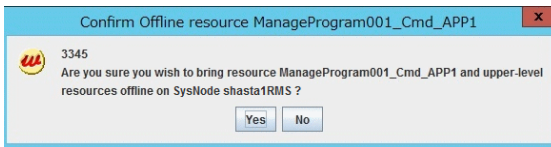


Figure 146: Confirming a resource stop

CLI: hvutil -f

The syntax for the CLI is as follows:

```
hvutil -f gResource
```

This is a normal offline request. There is no "forced" offline request for gResource.

Note

- Stopping a resource is possible only when the application which controls the resource is Online.

- If some resource becomes `Online` unexpectedly while the `userApplication` is partially `Online`, resource stop request is rejected on the node.
- When a resource fails to be stopped, if the resource, which is not a target resource to be stopped, remains in the `Online` state, the application becomes the `Inconsistent` state. To restore this state, remove the cause of the unexpected stop of the resource, and then, use `hvutil -f` to stop `userApplication`. After that, start the resource if necessary.

8.3.4 Notes on starting and stopping a resource

- The information, on which only some resources under `userApplication` are being started, is synchronized on all the nodes where this `userApplication` operates. Therefore, if an application is switched automatically while only some resources are being started, the only resource that was started on the source node can be started on the remote node.
- If some resource becomes `Online` unexpectedly while the `userApplication` is partially `Online`, the `userApplication` remains in `Online` state (does not become `Inconsistent` state). If an application is switched automatically in this state, the unexpected started resource cannot become `Online` on the remote node.
- To start and stop the resources under a `userApplication` that is controlled by the scalable controller, the controlled `userApplication` must be `Offline`.



To stop the unexpected started resource, make sure to change the corresponding `userApplication` to the `Maintenance` mode, and then stop the target resource manually (if the resource is stopped without changing the mode to `Maintenance` mode, the resource is considered to be a resource error).

8.3.5 Fault Traces of Resources

When a resource fault occurs, you can check which resource has a fault from the RMS main window of Cluster Admin.

- The following fault trace icon (⚠️) is displayed in the right side of the state icon of the failed resource in the cluster tree (Figure 147).

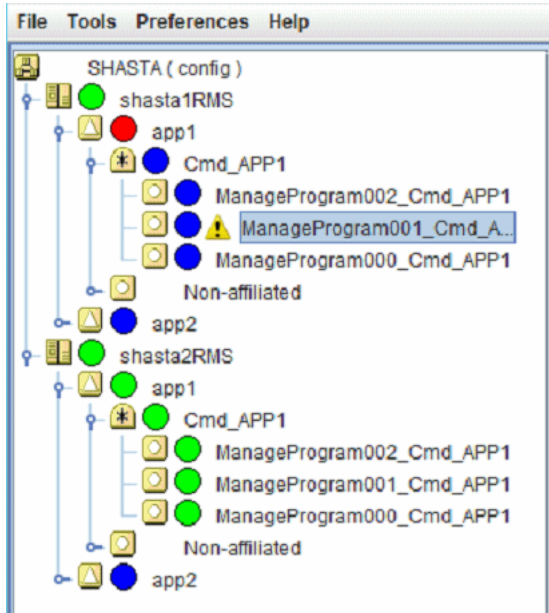


Figure 147: Checking the failed resource



Note

- The fault trace icon shows the resource error has occurred in the past. It does not indicate the current state of the resource. If you want to check the current state of the resource, check the state of the resource object.
- The fault trace icon is displayed in the resource that an error is notified to RMS. Normally, the fault trace icon is displayed on the failed resource. However, the superior resource in the graph tree may detect the failure faster than the actual failed resource depending on the monitoring timing. For example, when the failure occurs in the file system resource, normally the failure is notified to Fsystem resource. Depending on the monitoring timing, however, the superior resource in the graph tree may detect the failure faster than Fsystem resource detects the failure. This can

be found when an operation application that accesses the file system is registered as a resource and the operation application first detects the failure. In this case, the fault trace icon may be displayed for only the superior resource.

- If a resource error is notified from another resource during the state transitions, the resource may not be displayed with the fault trace icon.
- **Fault Occurred** is displayed in the `StateDetails` attribute of the failed resource (Figure 148).

RMS Attribute	Value
<code>StateDetails</code>	Fault Occurred
<code>MonitorOnly</code>	0
<code>NonCritical</code>	0
<code>Operator</code>	AND
<code>AutoRecover</code>	0
<code>FaultRecoverable</code>	0
<code>StandbyCapable</code>	1
<code>Class</code>	CommandLine/Arbitrary
<code>ClusterExclusive</code>	1
<code>DeviceName</code>	ManageProgram001_Cmd_APP1
<code>LieOffline</code>	0
<code>rKind</code>	0
<code>rName</code>	ManageProgram001_Cmd_APP1
<code>RestartOnMod</code>	1
<code>NoDisplay</code>	0
<code>Affiliation</code>	Cmd_APP1

Figure 148: Displaying Fault Occurred in the `StateDetails` attribute

CLI: `hvdisp -a`

The syntax for the CLI is as follows.

```
hvdisp -a
```

Fault Occurred is displayed in the `StateDetails` attribute of the failed resource.

Clear the fault trace of the resource with the following procedure.

- ▶ Right-click on the failed resource, and then select [Clear fault trace (`hvtutil -c`)] from the pop-up context menu (Figure 149).

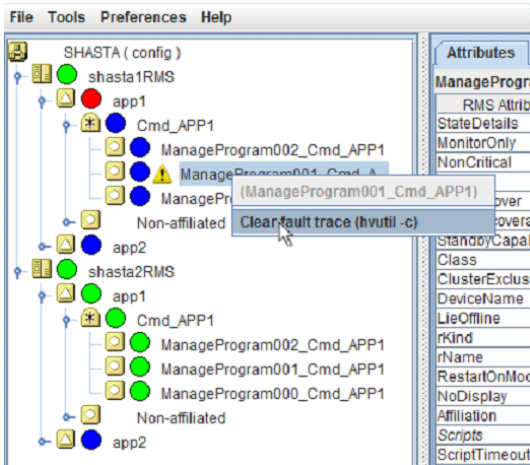


Figure 149: Clearing the Fault trace

- ▶ Before the command is executed, you are prompted to confirm whether the operation can be performed. Click *Yes* to start to clear the fault trace (Figure 150).

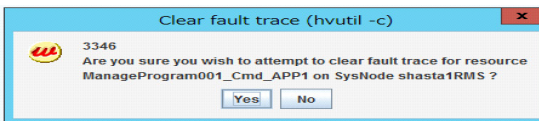


Figure 150: Checking whether the Faulted trace is cleared

CLI: hvutil -c

The syntax for the CLI is as follows:

```
hvutil -c gResource
```

i In addition to the `hvutil -c` command can clear the fault trace, it can be also cleared automatically when the resource becomes `Online` next time. Once the fault trace is cleared, its icon will be disappeared and "Fault Occurred" will not be also displayed in the `StateDetails` attribute.

8.4 Using maintenance mode

Maintenance mode is a special mode of operation that allows an application to be temporarily decoupled from its dependent resources. This allows, for example, a file system to be taken offline for backup purposes without disrupting the online state of its parent application.

8.4.1 Entering maintenance mode

You can enter maintenance mode for all applications on all the nodes as follows:

- ▶ Right-click on the cluster at the top of the RMS tree and select *Enter Maintenance Mode* from the popup menu (Figure 151).

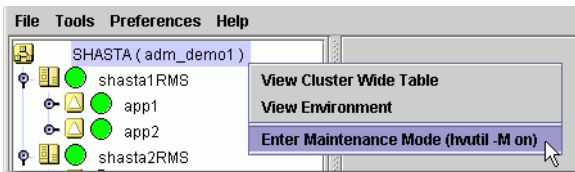


Figure 151: Starting maintenance mode for all applications

Enter maintenance mode for only one application as follows:

- ▶ Right-click on an application instance in the RMS tree and select *Enter Maintenance Mode* from the popup menu (Figure 152).

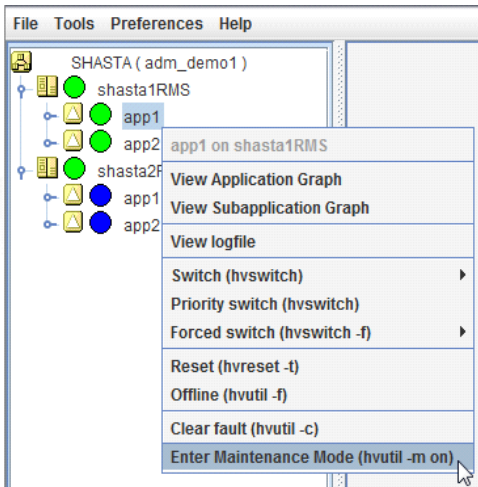


Figure 152: Starting maintenance mode for a single application

In either case, you will be prompted to confirm the operation (Figure 153 and Figure 154).

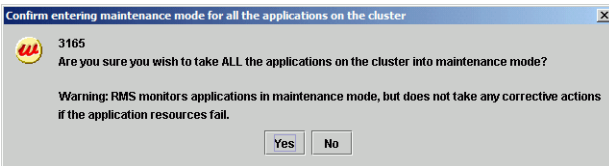


Figure 153: Maintenance mode confirmation for all applications

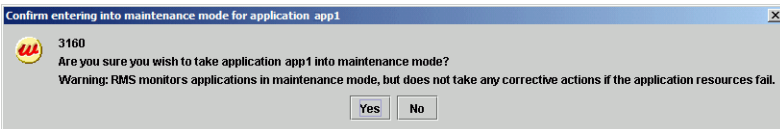


Figure 154: Maintenance mode confirmation for one application



Maintenance mode is clusterwide: if an application is in maintenance mode on one node, it is also in maintenance mode on any other node where it can run.

The application is allowed to start in the following states.

- Online state
- Standby state
- Offline state
- Warning state
- Inconsistent state (If an application state just before it becomes inconsistent is Faulted state, it is excluded.)

The maintenance mode is rejected if there is any request of dynamic reconfiguration of RMS graph.

Figure 155 shows the Cluster Admin window after one application is put into maintenance mode.

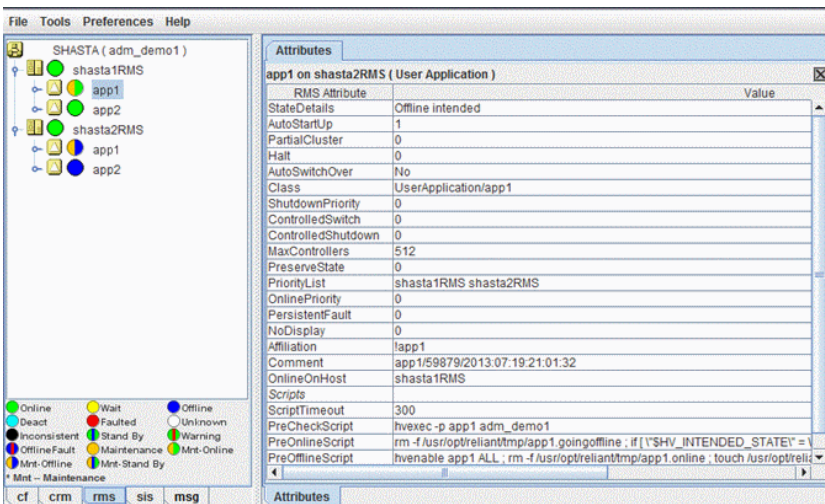


Figure 155: Typical cluster in maintenance mode

Note how the right half of the application status icons indicate the intended status (the status just before the start of maintenance mode). The intended status is also indicated by the application's *StateDetails* attribute, which is the first item in the *Attributes* table in the right pane.

8.4.2 Maintenance mode operating notes

When an application enters maintenance mode (MM), it affects all other applications that **share the same graph**. For example, if two applications are linked by a controller, then putting one in maintenance mode will cause the other to go into maintenance mode as well; which one is the parent and which one is the child does not matter in this case.

Conversely, if two applications do not share the same graph, *i.e.*, they are not linked by one or more controllers, then one can be put into MM while the other operates under normal RMS control.

For instance, in the example presented earlier, `app1` and `app2` are independent. While `app1` is in MM, `app2` continues to operate normally and can be switched from one node to the other, as illustrated in Figure 156 below.

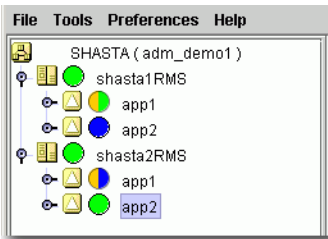


Figure 156: Normal operation of independent application


8.4.2.1 Overall cluster restrictions in maintenance mode

Even though some applications may continue to operate under normal RMS control, MM still places restrictions on the overall cluster operation. In particular, note the following:

- You must exit MM before starting or stopping an application or a resource.
- You must exit MM everywhere in the cluster before you can shut down RMS.

8.4.3 Exiting maintenance mode

To exit maintenance mode, use the following procedure:

- ▶ Confirm that there is no  indicated to the right side of the application icon. If there is, return all the resources under `userApplication` to the status just before the start of maintenance mode.

The status just before the start of maintenance mode can be determined with the color of the application's icon. For details on the icon's color, see "7.1.3 RMS Main Window" of "PRIMECLUSTER Installation and Administration Guide."

- ▶ Right-click on the cluster or an application and select *Exit Maintenance Mode* from the popup menu (Figure 157 and Figure 158).

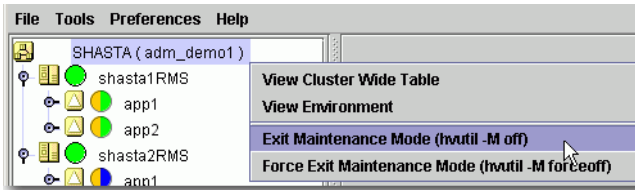


Figure 157: Normal maintenance mode exit for all applications

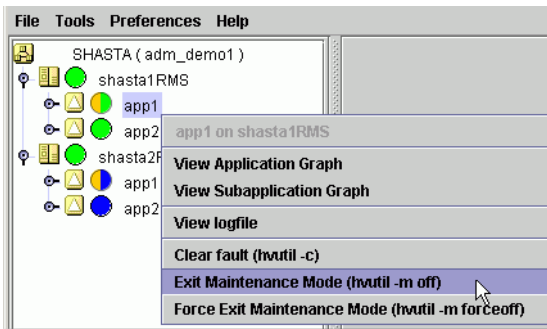


Figure 158: Normal maintenance mode exit for a single application

In either case, you will be prompted to confirm your action before the operation proceeds.

i You can exit maintenance mode for a single application even if you entered maintenance mode for the entire cluster.

To exit maintenance mode, the following conditions must be met:

- If you start maintenance mode when the application status is either Online, Offline, Standby, or Warning, statuses of the application and each resource must be the same as those before maintenance mode is started in order to exit maintenance mode. If the status is different, you cannot exit maintenance mode.

- If you start maintenance mode when the application status is `Inconsistent`, remove causes which put the application into the `Inconsistent` state. You cannot exit maintenance mode until causes are removed.

For example, in the above example, there is a file system resource `Res1` in the application `app1`, and `Res1` is in the `online` state on `shasta2RMS` when starting the maintenance mode in `app1`. In this case, `Res1` must be in the `online` state on `shasta2RMS` to exit the maintenance mode of `app1`.

If you forcibly exit the maintenance mode by using [`Force Exit Maintenance Mode`] in a different state from when starting the maintenance mode, the application enters the `Inconsistent` state. Therefore, we do not recommend using [`Force Exit Maintenance Mode`].

Note that both the cluster and application popup menus shown above contain a *Force Exit Maintenance Mode* item. If you choose this command, it will force RMS to exit maintenance mode even if some resources are not in the appropriate state. The prompt to confirm the operation for one application is shown in Figure 159; the prompt for all applications is similar.

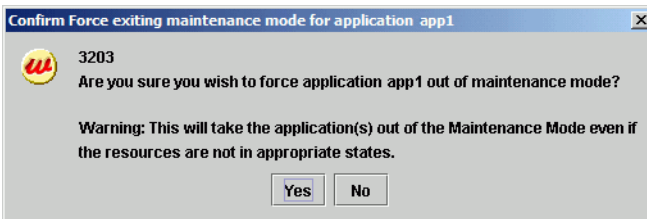


Figure 159: Forced maintenance mode confirmation for all applications

If the status of `userApplication` before the start of maintenance mode is `Online` or `Standby`, `Online` or `Standby` processing is executed during the exit of maintenance mode.

In these operations, if resource statuses under `userApplication` are the same as those before maintenance mode is started, the `Online` script is not executed and the resource statuses are not changed.

However, if the `Online` processing is executed for `Cmdline` resource to which the `NULLDETECTOR` attribute is set, the `Online` script is definitely executed despite the actual resource statuses.

8.4.4 Maintenance mode CLI: `hvutil -m` and `-M`

Control maintenance mode (MM) with the `hvutil` command:

```

hvutil -M { on | off | forceoff }
hvutil -m { on | off | forceoff } userApplication

```

Options:

-M Applies the MM operation to all applications on all the nodes

-m Applies the MM operation to the specified application on all the nodes

Operations:

on Starts maintenance mode

off Stops MM if all resources are in the appropriate state

forceoff Forces MM to stop even if all resources are not in the appropriate state

The `hvutil` maintenance mode commands operate synchronously, so they do not return until the final state has been reached or until an error occurs. In the particular case where '-m off' returns a failure because one or more resources were in an inappropriate state, an error message is displayed that lists the problem resources.

**Note**

To stop the maintenance mode, confirm that all the resources under `userApplication` are in the status just before the start of maintenance mode.

If one of the following information is displayed in `StateDetails` column of the targeted `userApplication` in the output of `hvdisp` command, there is a need to return the `userApplication` status to the status just before start of maintenance mode.

- "Online!!" is being displayed (Online is the status just before starting maintenance mode).
- "Standby!!" is being displayed (Standby is the status just before starting maintenance mode).
- "Offline intended" is being displayed and one or more resources under the `userApplication` is neither Offline status nor Standby status (Offline is the status before starting maintenance mode).

9 Appendix—Site preparation

"Software Release Guide" and "PRIMECLUSTER Installation Guide" for your operating system describes how to prepare your cluster to operate RMS. Some of the procedures require you to modify system files so that RMS can identify the hosts, file systems, and network interfaces used in a configuration. You should have completed these procedures when RMS was installed.

In some cases, you will be creating or modifying your RMS configuration because changes have been made to your site. Certain site changes may require you to review and update your system files first. These changes include, but are not limited to, the following:

- IP addresses were changed.
- Redundant interconnects were added to the cluster.
- Hosts were added, removed, or renamed.
- Two or more clusters were merged into one.
- File systems or SANs were added or removed.

For convenience, the site preparation descriptions for hosts, file systems, and networks are duplicated here. If any of these specifications have changed since your initial RMS installation, you should review this material and make the necessary adjustments before proceeding with your RMS configuration.

The modifications generally involve adding RMS-specific entries to standard system files; pre-existing entries required for proper operation of your hosts and network are not affected. Resources for market-specific applications may require similar customization.

9.1 Network database files

9.1.1 /etc/hosts

The `/etc/hosts` file must contain the IP addresses and RMS names of all the host systems that are part of the cluster.

RMS uses its own internal set of host names to manage the machines in the cluster. When you configure the cluster, you will use the RMS host names and not the standard host names. These names must be entered in `/etc/hosts` on each system in the cluster to avoid problems should access to the DNS fail. If you used Cluster Admin to configure CIP for RMS, then `/etc/hosts` will already contain the correct RMS node names described below.

By default, the names follow the conventions in Table 6.

Entry type	RMS naming pattern	Examples
RMS host name	<code><hostname>RMS</code>	shasta1RMS shasta2RMS

Table 6: RMS host name conventions in `/etc/hosts`



The RMS host name for a machine must match the contents of the `RELIANT_HOSTNAME` variable in that machine's `hvenv.local` configuration file, if that file exists.

RMS does not support IPV6 addresses.

Example

The following entries in `/etc/hosts` are for a cluster with nodes `shasta1` and `shasta2`. The interface names are assigned as follows:

- Standard host names on the public network 172.25.220
- RMS node names on the private network 192.168.10

```
172.25.220.83  shasta1
172.25.220.84  shasta2
# node names for RMS
192.168.10.83  shasta1RMS
192.168.10.84  shasta2RMS
```

9.1.1.1 Network interface names in `/etc/hosts`

If you plan to configure one or more network interfaces for switchover with the *Ip Address* subapplication, you must first enter the interface name(s) in the `/etc/hosts` file on every node where that interface can exist. Each entry consists of the interface IP address and its name in the normal format; no special comments are required.

Example

An example is shown below.

- If you switch the interface `shastavip` with IPv4 address
`172.25.222.223:`
`172.25.222.223 shastavip`
- If you switch the interface `shastavip6` with IPv6 address
`fd00:1::219:99ff:fe18:1b4e:`
`fd00:1::219:99ff:fe18:1b4e shastavip6`



When you configure the *Ip Address* subapplication, you specify the interface name as it appears in `/etc/hosts`, and not the IP address.

- IPv6 link local addresses are not available.
- When defining the interface name in the `/etc/hosts` file, do not assign the same interface name to the IPv4 and IPv6 addresses.

9.1.2 `/root/.rhosts` (Linux) and `/.rhosts` (Solaris)

Contains entries to control trusted login from remote hosts.

The Wizard Tools require automatic login or authentication as `root` on every machine in the cluster. One method is to include the names of trusted hosts in the `.rhosts` file, which must be modified appropriately on each node. See the `rhosts` manual page for a complete description of the format.

Example

If the cluster consists of hosts `shasta1` and `shasta2`, then every machine's `.rhosts` file should contain the following lines:

```
shasta1 root  
shasta2 root
```




The Cluster Foundation (CF) provides the equivalent of `.rhosts` functionality for all RMS configuration, administration, and operation tasks.

9.2 Configuration resource definitions

9.2.1 /opt/SMAW/SMAWRrms/etc/hvipalias

This file contains entries for all of the network interfaces that are to be used as resources in the configuration. Typically, each entry associates a logical interface name, or **IP alias**, with a physical interface on a specified node. The IP alias always presents the same IP address, even though it is switched from node to node, and even if the underlying physical interface has different characteristics on each node.

 Each IP alias with its IP address must be entered in the `/etc/hosts` file. The IP address does not appear in the `hvipalias` file.


Each entry in `hvipalias` must contain the following fields:

`<Uname> <IfName> <IfDevice> <Netmask/Prefix>`

The fields are defined as follows:

- *Uname*—Name of the machine to host the logical interface. This is usually the value returned by the `'uname -n'` command.

Alternatively, you can use the Cluster Foundation (CF) node name, which is returned by the `'cftool -q1'` command. This can be used to differentiate two machines which are configured with the same `'uname -n'` setting, provided they were assigned different names when CF was configured.
- *IfName*—Logical interface name, or IP alias. This name must appear with its associated IP address in the node's `/etc/hosts` file, and the associated IP address must be the same on every node.
- *IfDevice*—Physical device name to be associated with the logical interface when it is switched to the specified machine.
- *Netmask/Prefix*—The netmask (IPv4) or the prefix length (IPv6) to use with the IP address associated with the interface name.
The netmask (IPv4) must be set as the 8-digit hexadecimal number string. The prefix length (IPv6) must be set as the decimal number string between 0 to 128.

 To avoid network conflicts, a network hostname or address monitored by the *IpAddress* subapplication can be active on only one node in the cluster at any time.

When a configuration is activated, the local `hvipalias` file is copied to every other node in the configuration; the file with same name and path on the remote node will be overwritten with the new content. Therefore, the file must contain one line for every combination of interface and host where that interface can be switched.

For instance, if the interface named `dbhost` can be switched between two nodes, then `hvipalias` on each node should contain lines for both the local and remote interfaces:

Example (Linux)

```
#Uname   IfName   IfDevice  Netmask
shasta1  dbhost   eth1      0xffffffff00
shasta2  dbhost   eth1      0xffffffff00
```

Example (Solaris)

```
#Uname   IfName   IfDevice  Netmask
fuji2    dbhost   hme0      0xffffffff00
fuji3    dbhost   hme0      0xffffffff00
```

9.2.2 /opt/SMAW/SMAWRrms/etc/hvconsoles

Controls customized handling of fault messages, usually to remote consoles or special devices such as pagers. This does not affect the standard messages written to the RMS or system log files.

Each entry specifies a program to be executed when an RMS resource object encounters a fault.

If the file does not exist, you will receive no customized fault information. A complete description of the format is available in the comments in the `hvconsoles.template` file.

9.3 Linux file systems

To manage Linux file systems with RMS, you must create entries in the `/etc/fstab.pcl` configuration file as described in the following sections. This configuration file has the following features:

- Leading whitespace and empty lines are ignored.
- A line beginning with the string `'#RMS#' or '#RMS:<appname>#'` is a file system specification for RMS. The entire specification must be on the same line—continuation to additional lines is not allowed.
- Any other line beginning with a pound sign (`#`), or any line beginning with an asterisk (`*`), is treated as a comment.
- All other lines are presently ignored. However, they may be processed by future versions of RMS.

9.3.1 `/etc/fstab.pcl`

This file contains entries for all of the local and remote file systems that are to be used as resources in the configuration. RMS is responsible for mounting and unmounting each of these file systems in order to bring them online or offline, respectively, according to the requirements of the running configuration.

For each file system to be managed by RMS, create a line in `/etc/fstab.pcl` with the standard `fstab` fields, and then insert the string `#RMS#` at the beginning of the line. For more information, see the `fstab` manual page.

Note the following restrictions when you create `/etc/fstab.pcl`:

- Do not specify the same file system in both a standard `/etc/fstab` entry and an RMS `/etc/fstab.pcl` entry. The standard entry will mount the file system at system startup, and this will create a conflict when RMS starts up and attempts to mount the same file system.
- If a remote file system is specified in the form `<server_name>:<server_path>`, then `<server_name>` must be a host name that appears in the `/etc/hosts` file. It cannot be an IP address, and you should not rely on DNS to resolve the name.

Examples

```
#RMS#/dev/sdb2 /fs2    ext2    defaults 1 2
#RMS#/dev/sda1 /mnt/data1 auto    noauto,user 0 0
#RMS#/dev/sda2 /mnt/data2 auto    noauto,user 0 0
```

9.3.1.1 Configuring file systems for particular applications

If the RMS comment is of the form `#RMS:<appName>#`, the file system entry applies only to the specified application. From an RMS perspective, file systems assigned to a given application are independent of those assigned to other applications. A file system can be assigned to two or more applications, provided only one of the applications is online at any time.

Examples

```
#RMS:app1#/dev/sdb2 /data3 auto    noauto,user 0 0
#RMS:app2#/dev/sdb6 /data4 auto    noauto,user 0 0
```

9.3.1.2 Clusterwide configuration issues

In general, if you create an `/etc/fstab.pcl` control entry for a remote file system or a shared filer on one node, then you should duplicate that entry on every other node in the cluster, even if some nodes will not mount that file system. This helps to ensure that the configuration behaves consistently throughout the cluster.

Use a similar procedure for entries that specify local file systems and mount points. If all the nodes have the same architecture, you may be able to simply copy the entire `/etc/fstab.pcl` control file. However, if the local physical disk device differs from node to node, you must individually adjust the entries for the same mount point. For example, the respective entries for `/mnt1` on `node1` and `node2` might be as follows:

node1:

```
#RMS#/dev/sda3 /mnt1 ...
```

node2:

```
#RMS#/dev/sdb5 /mnt1 ...
```

In all cases, for each mount point that appears in `/etc/fstab.pcl`, be sure to create the directory on every node in the cluster.

9.4 Solaris file systems

- ▶ `/etc/vfstab.pcl`

Contains entries for all of the local file systems that are to be used as resources in the configuration. In other words, this file describes the file systems that should be mounted locally.

For each file system to be managed by RMS, create a line with the standard `vfstab` fields, and then insert the string `#RMS#` at the beginning of the line. RMS entries appear as comments and will be ignored by all processes other than PRIMECLUSTER components. For more information, see the `vfstab` manual page.

Example

```
#RMS#/dev/dsk/c0t0d0s0 /dev/rdk/c0t0d0s0 /testfs1 ufs 1 yes -
```

- ▶ `/etc/dfs/dfstab.pcl`

Contains entries for all of the shared remote resources in the high-availability configuration. In other words, this file describes the file systems that can be mounted on a remote node.

For each file system to be managed by RMS, create a line with the standard `dfstab` fields, and then insert the string `#RMS#` at the beginning of the line. RMS entries appear as comments and will be ignored by all processes other than PRIMECLUSTER components. Therefore, to ensure that the NFS daemons start at boot time, there must be at least one non-comment, non-RMS entry in this file.

The non-RMS entry might be a dummy entry configured for a local file system and shared only to the local node. This would mean that no real sharing to a remote node is done, but it would still cause the NFS daemons to be started. For more information, see the `dfstab` manual page.

Example

The following contains both a non-RMS entry and an RMS entry:

```
share -F nfs -o ro=localhost /var/opt/example
#RMS# share -F nfs -o rw,
root=shasta1RMS:shasta1:045nfs045dia1:045msg:shasta1RMS: /sapmnt/045
```

9.4.1 NFS Lock Failover

The NFS Lock Failover feature is available for local file systems. If you configure NFS Lock Failover for a file system and the file system subsequently fails, then both the file system and its NFS locks will failover to the same node.

This feature requires the following site preparation steps:

- You must have a shared disk accessible to all the nodes in the cluster.
- You must dedicate a directory to NFS Lock Failover on that shared disk. If you specify a directory that already exists, no other applications will be allowed to use it after it is configured.
- You must reserve one IP address for each application that uses NFS Lock Failover. You will specify this IP address when you configure the local file system for NFS Lock Failover. You will also configure this address in an Ip Address subapplication so it switches with the application that contains the file system.



Only one file system per userApplication object can be selected for NFS Lock Failover.

9.5 NFS servers

In a high availability environment such as RMS, an exported file system must be able to failover transparently when its server node is taken out of service: clients that mounted the file system before the failover should experience no access problems after the failover. NFS file systems require special preparation to achieve this result.

When a client mounts a remote NFS file system, it creates an internal **file handle** that it uses for future operations with the file system. To comply with NFS architecture, the client file handle includes the server's major and minor device numbers for the file system. This design can create access problems in the RMS environment. If the file system goes offline on the original server, and then comes back online on a second server that assigns different major and minor device numbers, the file handle will no longer be valid. This condition is called a **stale file handle**. The solution is to assign the same major and minor device numbers to the file system on every NFS server that may advertise that file system.

The above discussion refers to file systems in general, but in a high availability environment, the file system will actually be a shared disk volume that is accessible from any node that will export it. Preparing a shared disk volume with the same major and minor device number may require changes in the hardware or software configuration. If the shared disk volume is built on top of volume management software, additional steps may be necessary when the volume manager is installed.

This section provides some tips for preparing volume managers for use as NFS servers in the RMS environment.

9.6 Log files

9.6.1 `/var/log/messages` (Linux) or `/var/adm/messages` (Solaris)

By default, all RMS messages go to both the system log, `messages`, and the RMS `switchlog` file (located by default in `/var/opt/SMAWRrms/log`). If you do not want to send messages to the system log, then set `HV_SYSLOG_USE = 0` in the `hvenv.local` file. By default, `HV_SYSLOG_USE = 1`.

9.7 Other system services and databases

RMS requires the following system services or databases to be configured according to the instructions in "Software Release Guide" and "PRIME-CLUSTER Installation Guide" for your operating system:

- PRIMECLUSTER Cluster Foundation (CF), including CIP
- `/etc/nsswitch.conf` system service lookup order database
- `echo` service—Linux only

10 Appendix—States

10.1 Basic states

Table 7 lists the states that detectors may report to the base monitor:

State	Description
Faulted	Error condition encountered. The error may have occurred in the resource, in one of its children, or during script processing.
Offline	Disabled, not ready for use. The scripts have successfully disabled the resource.
Online	Enabled, ready for use. All required children are online, and no errors were encountered while scripts were processed.
Standby	Ready to be quickly brought Online when needed.

Table 7: States reported by detectors for RMS objects

Table 8 lists additional resource states that may be displayed in the Cluster Admin GUI or by `hvdisp`:

State	Description
Deact	Applies to <code>userApplication</code> objects only. Operator intervention has deactivated the application throughout the cluster (such as for maintenance purposes).
Inconsistent	Applies to <code>userApplication</code> objects only. The object is <code>Offline</code> or <code>Faulted</code> , but one or more resource objects in its graph have their <code>ClusterExclusive</code> attribute set to 1 and are <code>Online</code> or <code>Faulted</code> .
OfflineFault	Fault that occurred in the past has not yet been cleared.
Unknown	No information is available. Reported before object initialization is completed.

Table 8: Additional states that may be displayed for RMS objects

State	Description
Wait	Temporarily in transition to a known state. An action has been initiated for the affected resource, and the system is waiting for the action to be completed before allocating one of the above states.
Warning	Some warning threshold has been exceeded. Note that this state is reported only for selected resources. —
Maintenance	Manual, temporary mode of operation in which the state of an application is decoupled from the states of its dependent resources. This allows, for example, a file system to be taken offline for backup without disturbing the state of its parent application. An application in maintenance mode is usually marked with its intended status. The intended status is the status just before the start of maintenance mode. The maintenance mode intended statuses are <code>Maintenance-Online</code> , <code>Maintenance-Offline</code> , and <code>Maintenance-Standby</code> .

Table 8: Additional states that may be displayed for RMS objects

The interpretation of `Offline` and `Faulted` may depend on the resource type. For instance, a mount point resource can be either `Online` (mounted) or `Offline` (not mounted); in this case, the detector would never report the `Faulted` state. On the other hand, a detector for a physical disk can report either `Online` (normal operation) or `Faulted` (input or output error); it would never report `Offline`.

10.2 State details

Besides the basic states listed above, RMS may report additional state details in the following locations:

- In the Cluster Admin GUI, the properties view of an object includes the *State Details* item at the top of the list. Unlike most other attributes, which are determined at configuration time by the Wizard Tools, this information-only field is dynamically set by RMS at runtime.
- In the output of the `hvd disp` utility, the `StateDetails` column appears at the end of each line.

In most cases, the *StateDetails* field is empty. RMS typically provides this extra information when an application is in maintenance mode, or when an object is in a transitional, inconsistent, or standby state. Table 9 lists all possible *StateDetails* values for RMS objects.

Value	Description
Failed Over	Offline processing successful and failover initiated
Faulted	Received Faulted report
Fault Occurred	The resource error has occurred in the past. This value is displayed until the Fault trace is cleared.
Inconsistent on remote	userApplication is Online on multiple hosts, but is not Online on the local host
Initial Fault	userApplication already faulted when RMS started
Joined	SysNode is in Offline state because it has joined the cluster
Killed	SysNode is in Faulted state because of a successful kill
Not Joined	SysNode is in Offline state because it has not yet joined the cluster
Offline	Received Offline report
Offline Failed	Offline processing failed
Offline Success	Offline processing successful
Offline intended	Intended state is Offline
Online	Received Online report

Table 9: StateDetails values for RMS objects

Value	Description
Online	userApplication is Online on multiple hosts
Online !!	Intended state is Online, but some resources under userApplication are not Online
Online intended	Intended state is Online
Partial	Resources under the userApplication are partially started
PreCheckScriptFailed	PreCheckScript failed
Preserved	PreserveState set, no Offline processing initiated
Shutdown	SysNode is in Faulted state because it has been shutdown
Standby	Received Standby report
Standby !!	Intended state is Standby, but some resources under userApplication are not Standby
Standby intended	Intended state is Standby

Table 9: StateDetails values for RMS objects

For example, if an application was online on a particular node before it was put into maintenance mode, it will generally return to the online state on the same node when it leaves maintenance mode. RMS indicates this by reporting *Online intended* in the state details field on that node. On other nodes where the application was previously offline, RMS will report *Offline intended* in the state details field.

11 Appendix—Object types

The following alphabetical list describes all object types that are supplied with RMS and configured by the Wizard Tools.

andOp

Required attributes:

HostName (for direct children of a userApplication object)

Object associated with its children by a logical *AND* operator. This object type is online if all children are online, and offline if all children are offline.

controler

Required attributes:

Resource

Object that allows a parent userApplication to control one or more child userApplication objects.

ENV

Required attributes:

(none required)

Object containing clusterwide (global) environment variables.

ENVL

Required attributes:

(none required)

Object containing node-specific (local) environment variables.

gResource

Required attributes:

rKind

rName

Custom (generic) object. Usually represents system resources such as file systems, network interfaces, or system processes.

orOp

Required attributes:

(none required)

Object associated with its children by a logical *OR* operator. This object type is online if at least one child is online.

SysNode

Required attributes:

(none required)

Represents nodes in the cluster; at least one required. Only `userApplication` objects are allowed as its children.

userApplication

Required attributes:

(none required)

Represents an application to be monitored; at least one required. Must have one or more `SysNode` objects as its parents. For each `SysNode` parent, it must have one child `andOp` with its `HostName` attribute set to the name of the corresponding `SysNode`.

12 Appendix—Attributes

Some object types require specific attributes for RMS to monitor that object type. Some attributes can be modified through the user interface, while others are managed internally by the Wizard Tools. The following sections describe the possible values of all attributes. The default value varies depending on each resource.

12.1 Attributes available to the user

Attributes in this section can be changed using the Wizard Tools user interface.

AutoRecover

Possible Values: 0, 1

Valid for resource objects. If set to 1, executes the online script for an object if the object becomes faulted while in an `Online` state. If the object is able to return to the `Online` state, the fault is recovered.

This attribute must be 0 for `controller` objects: RMS handles switchover of child applications automatically.

AutoStartUp

Possible Values: yes, no

Valid for `userApplication` objects. If set to "yes," automatically brings the application `Online` when RMS is started.

You can override the `AutoStartUp` attribute for all `userApplication` objects by setting the `HV_AUTOSTARTUP` variable. See the description of `HV_AUTOSTARTUP` in the "Local environment variables".

AutoSwitchOver

Possible Values: Valid string containing one or more of the following: No, HostFailure, ResourceFailure, ShutDown

Valid for `userApplication` objects. Configures an application for automatic switchover if it becomes faulted. The values can be combined using the vertical bar ("|") character. The `No` value inhibits automatic switchover and cannot be combined with any other value.

For backward compatibility, the numeric values 0 and 1 are accepted: 0 is equivalent to `No`, and 1 is equivalent to `HostFailure | ResourceFailure | ShutDown`.

ClusterExclusive

Possible Values: 0, 1

Valid for resource objects. If set to 1, guarantees that the resource is online on only one node in the cluster at any time. If set to 0, allows a resource to be online on more than one node at a time. Note that “online” in this context refers to any phase of online processing. For instance, if a resource is in the `Online` state on one node while its `PreOnlineScript` is executing on another node, then both resource objects would be considered as online for the purposes of this test.

The user can modify this attribute for `cmdline` subapplications only. The configuration tools control this attribute for all other subapplications.

FaultScript

Possible Values: Valid script (character)

Valid for all object types. Specifies a script to be run if the associated object enters the `Faulted` state.

- The argument specified during registration of script will be specified as it is during execution of script.
- Wait until this script finishes in the `Fault` processing.
- If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file but the `Fault` processing continues.

For details about `Fault` processing, refer to "Fault processing".

HaItFlag (HaIt)

Possible Values: no(0), yes(1)

Valid for `userApplication` objects. Controls local node elimination in the event of a double fault. A double fault occurs when a second fault is generated during the initial fault processing of an application.

If `HaItFlag` is set to `yes`, and another node is available to run the application, a double fault will trigger the following sequence of events:

1. First, RMS on the local node will exit immediately.
2. Next, RMS on another node will invoke the Shutdown Facility to eliminate the local node.
3. Finally, all applications that were online on the local node, and that have their `AutoSwitchOver` parameter set to include `HostFailure`, will be switched over to the available node.

Note: Even if all the conditions for the `HalTFlag` attribute are met for an application (`AutoSwitchOver` setting, additional hosts available), other applications running on the same host may block the `HalTFlag` operation. For instance, another application may have no other available hosts, or it may not have the appropriate `AutoSwitchOver` setting. In either case, RMS will continue to run on the local node. To prevent this, allocate additional hosts for the other applications and adjust their priority lists to minimize node conflicts with the application that has its `HalTFlag` attribute set.

`LieOffline`

Possible Values: 0, 1

Valid for all resource objects. If set to 1, allows the resource to remain `Online` during `Offline` processing.

`MonitorOnly`

Possible Values: 0, 1

Valid for resource objects. If set to 1, `userApplication` does not become `Faulted` and switching does not occur even if that resource becomes `Faulted`.

If the resource becomes `Faulted` during `Offline` processing due to a manual switchover, for example, `Offline` processing continues but the switchover is canceled.

`OfflineDoneScript`

Possible Values: Valid script (character)

Valid for `userApplication` objects. Specifies the script to be run after the `Offline` processing of `userApplication` has completed.

- The argument specified during registration of script will be specified as it is during execution of script.
- Do not wait for this script to finish in the `Offline` processing.
- If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file but the `Offline` processing continues.

For details about `Offline` processing, refer to "Offline processing".

`OfflineScript`

Possible Values: Valid script (character)

Valid for all object types except `SysNode` objects. Specifies the script to be run to bring the associated resource to the `Offline` state.

OnlinePriority

Possible Values: 0, 1

Valid for `userApplication` objects. Allows RMS to start the application on the node where it was last online when the entire cluster was brought down and then restarted. If set to 0 or not set (the default), the application comes online on the node with the highest priority in the attribute `PriorityList`. If set to 1, the application comes online on the node where it was last online. In case of `AutoStartUp` or a priority switch, this last-online node has the highest priority, regardless of its position in the priority list.

RMS keeps track of where the application was last online by means of timestamps. The node which has the latest timestamp for an application is the node on which the application will go online. Different cluster nodes should be in time-synchronization with each other, but this is not always the case. Since RMS does not provide a mechanism for ensuring time-synchronization between the nodes in the cluster, this responsibility is left to the system administrator. If RMS detects a severe time-discrepancy between the nodes in the cluster, an `ERROR` message is printed to the switchlog.

`NTPD` or `CHRONY` can be used to establish consistent time across the nodes in the cluster. Refer to the manual page for `xntpd` or `chronyd` for more information.

The `OnlinePriority` persistent state information will be cleared if RMS is restarted with the last online node removed from the configuration.

OnlineScript

Possible Values: Valid script (character)

Valid for all objects except `SysNode` objects. Specifies the script to bring the associated resource to the `Online` or `Standby` state.

PartialCluster

Possible Values: 0, 1

Valid for `userApplication` objects. Specifies whether an application can negotiate online requests.

If set to 0, then the application can negotiate its online request only when all the nodes where it can possibly run are online.

If set to 1, then the application can negotiate its online request within the current set of online nodes, even if some other nodes (including the application's primary node) are offline or faulted.

PersistentFault

Possible Values: 0, 1

Valid for `userApplication` objects. If set to 1, when RMS is starting, the state of `userApplication` that stopped last time will be checked. Under any one of the following conditions, the `userApplication` becomes the `Faulted` state (`InitialFault`) right after RMS is started.

- The shutdown processing of RMS is not performed.
- `userApplication` is in the `Faulted` state when RMS is stopped.

If `userApplication` becomes the `Faulted` state, remove the cause of this error first, and then, use `hvutil -c` to clear the `Faulted` state.

**Note**

If the `PersistentFault` attribute is not set to 1, the `userApplication` does not become the `Faulted` state at RMS startup after a node error or panic reboot of the operating system. In this case, as it is not the `Faulted` state, this node is accepted as it can be switched or failed back. If the error definitely occurs on all the nodes that constitute cluster during the Online processing and also panic reboot of the operating system occurs, the failover may repeat infinitely.

PostOfflineScript

Possible Values: Valid script (character)

Valid for all objects except `SysNode` objects. Specifies the script to be run after the state of the associated resource changes to `Offline`.

PostOnlineScript

Possible Values: Valid script (character)

Valid for all objects except `SysNode` objects. Specifies the script to be run after the state of the associated object changes to `Online` or `Standby`.

- The argument specified during registration of script will be specified as it is during execution of script.
- Wait until this script finishes in the `Online` processing.
- If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file and the `Fault` processing will start.

For details about `Online` processing, refer to "Online processing".

PreCheckScript

Possible Values: Valid script (character)

Valid for `userApplication` objects. Specifies the script to be run before the start of `Online` or `Standby` processing.

- The argument specified during registration of script will be specified as it is during execution of script.
- If the script returns with an exit code zero, the `Online` or `Standby` processing will be started. If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file and the `Online` or `Standby` processing will not be started.

For details, refer to "PreCheckScript".

PreOfflineScript

Possible Values: Valid script (character)

Valid for all objects except `SysNode` objects. Specifies the script to run before the object is taken to the `Offline` state.

- The argument specified during registration of script will be specified as it is during execution of script.
- Wait until this script finishes in the `Offline` processing.
- If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file but the `Offline` processing continues. `Fault` processing will start after the `Offline` processing is completed.

For details about `Offline` processing, refer to "Offline processing".

PreOnlineScript

Possible Values: Valid script (character)

Valid for all objects except `SysNode` objects. Specifies the script to be run before the associated object is taken to the `Online` or `Standby` state.

- The argument specified during registration of script will be specified as it is during execution of script.
- Wait until this script finishes in the `Online` processing.
- If the script returns with an exit code value other than zero, the message indicating script failure will be recorded in switchlog file and the `Fault` processing will start.

For details about `Online` processing, refer to "Online processing".

PreserveState

Possible Values: 0, 1

Valid for `userApplication` objects. If set to 1, the resources are not to be taken `Offline` after a fault. Ignored if `ResourceFailure` is set to `AutoSwitchOver`.

ScriptTimeout

Possible Values: 0–`MAXINT` (in seconds) or valid string of the form "`timeout_value[:[offline_value][:online_value]]`"

Valid for all object types. Specifies the timeout value for all scripts associated with that object in the configuration file. RMS sends a kill signal to the script if the timeout expires.

Use the string format to specify individual timeout values of `offline_value` for `OfflineScript` and `online_value` for `OnlineScript`.

ShutdownPriority

Possible Values: 0–20

Valid for `userApplication` objects. `ShutdownPriority` assigns a weight factor to the application for use by the Shutdown Facility.

When interconnect failures and the resulting concurrent node elimination requests occur, SF calculates the shutdown priority of each subcluster as the sum of the subcluster's SF node weights plus the RMS `ShutdownPriority` of all online application objects in the subcluster. The optimal subcluster is defined as the fully connected subcluster with the highest weight.

StandbyCapable

Possible Values: 0, 1

Valid for resource objects. If set to 1, the object performs standby processing on all the nodes where the parent application is supposed to be `Offline`.

The user can modify this attribute for a `cmdLine` subapplication only. The configuration tools control this attribute for all other subapplications.

StandbyTransitions

Possible Values: `StartUp`, `SwitchRequest`, `ClearFaultRequest` or any combination joined by vertical bars (|)

Valid for `userApplication` objects. The value specifies when standby processing is initiated for the application object:

- `StartUp`—at startup. This setting is ignored if the real-world application is already online, or if the application object is forced to go online because the `AutoStartUp` attribute is set.
- `SwitchRequest`—after application switchover, if the application was online before the switchover.
- `ClearFaultRequest`—after a faulted state is cleared with `'hvutil -c'`.

WarningScript

Possible Values: Valid script (character)

Valid for resource objects with detector. Specifies the script to be run after the posted state of the associated resource changes to `Warning`.

12.2 Attributes managed by configuration wizards

Attributes in this section are managed internally by the configuration wizards or by RMS at runtime.

AlternateIp

Possible Values: Any interconnect name

Valid for `SysNode` objects. Space-separated list that RMS uses as additional cluster interconnects if the interconnect assigned to the `SysNode` name becomes unavailable. All these interconnects must be

found in the `/etc/hosts` database. By default, the configuration wizards assume the alternate interconnects to node `<nodename>` have names of the form `<nodename>rmsAI<nn>`, where `<nn>` is a two-digit, zero-filled number. This setting is restricted to very specific configurations and must never be used in a cluster with CF as interconnect.

Affiliation

Possible Values: Any string

Valid for resource objects. Used for display purposes in the user interface—no functional meaning within RMS.

AutoRecoverCleanup

Possible Values: 0, 1

Valid for `controller` objects. If set to 1, and `AutoRecover` is 1, then a faulted child application is requested to go `Offline` before recovering. If set to 0 and `AutoRecover` is 1, then a faulted child application recovers without going `Offline`.

Class

Possible Values: any string

Valid for all objects except `SysNode`. Describes the class of the resource object. Used by other programs for various purposes (for example, SNMP agents). This value is supplied by the configuration wizards.

Comment

Possible Values: any string

Valid for all objects. Used for documentation in the configuration file—no functional meaning within RMS.

ControlledSwitch

Possible Values: 0, 1

Valid for controlled `userApplication` objects. If set to 0, RMS allows a manual switch request from the CLI or the GUI. If set to 1, only the parent controller can issue switch requests to this `userApplication`.

DetectorStartScript

Possible Values: Any valid detector start script

Valid for resource objects with detector. Specify the detector start command directly in the `<configname>.us` file.

Note that a `controller` object has no detector because RMS determines its state internally.

HostName

Possible Values: Any SysNode name

Must be set only in the first-level andOp children of a userApplication object. Each of these andOp objects associates its parent application with the SysNode specified in its HostName attribute; the child andOp objects also determine the priority of the application's nodes.

I_List

Possible Values: Space-separated list of SysNode names

Valid for all SysNode objects. List of additional cluster interconnects that should be monitored by RMS. These interconnects are used only by customer applications and not by any PRIMECLUSTER products. All monitored interconnects must be found in the /etc/hosts database. In addition, all SysNode objects must have the same number of additional interconnects.

LastDetectorReport

Possible Values: Online, Offline, Faulted, Standby

Valid for resource objects with detector. This attribute contains the most recent detector report for the object. The value may be displayed in the Cluster Admin GUI; the possible values depend on the type of resource the object represents.

MaxControllers

Possible Values: 0–512

Valid for userApplication objects. Upper limit of parent userApplication objects for the specified child application.

NoDisplay

Possible Values: 0, 1

Valid for all object types. The resource object set to 1 is displayed only when specifying it with the -i option of the hvdisp command.

NullDetector

Possible Values: on, off

Valid for resource objects with detector. Used to disable a detector at runtime by setting NullDetector to on. This attribute is for use with dynamic reconfiguration only. NullDetector must never be set hard-coded to on in the RMS configuration file.

PriorityList

Possible Values: Valid list of SysNode names (character)

Valid for userApplication objects. Contains a list of SysNode objects where the application can come Online. The order in the list determines the next node to which the application is switched during a priority switchover, ordering a switchover after a Fault. The list is processed circularly.

The user specifies this attribute indirectly when selecting the nodes for an application. RMS uses the order in which the nodes were selected and creates PriorityList automatically. The user can change the PriorityList by adding individual nodes from the list in the desired order, rather than automatically selecting the entire list.

For applications controlled by or a controller object, the order of nodes in PriorityList is ignored. However, each child application must be able to run on the nodes specified for the parent application.

Resource

Possible Values: Valid name (character)

Valid for controller objects. Contains the name of the child (controlled) userApplication.

rKind

Possible Values: 0–2047

Valid for gResource objects. Specifies the kind of detector for the object.

rName

Possible Values: Valid string (character)

Valid for gResource objects. Specifies a string to be forwarded to the generic detector.

SplitRequest

Possible Values: 0, 1

Valid for controller objects. If set to 1, then PreOffline and PreOnline requests will be propagated to child applications separately from the Offline and Online requests. If 0, then separate PreOffline or PreOnline requests will not be issued for the child applications.

StateDetails

Possible Values: Any string

Valid for all objects. Displays additional state details in the Cluster Admin GUI or the `hvdisp` CLI user interface. In most cases, the state details field is empty. RMS typically provides this extra information when an application is in maintenance mode, or when an object is in a transitional, inconsistent, or standby state.

13 Appendix—Environment variables

This appendix provides a complete list of the environment variables used by RMS, grouped into the following types:

- "Global environment variables"
- "Local environment variables"
- "Script execution environment variables"

The discussion in the "Environment variables" describes how RMS manages environment variables.

13.1 Setting environment variables



Caution

Do not explicitly set RMS environment variables in the user environment. Doing so can cause RMS to lose environment variables settings.



Do not change the `hvenv` configuration file. Changes to your configuration's environment variables should be confined to the `<RELIANT_PATH>hvenv.local` file.

The values of environment variables are specified as `export` directives in the `hvenv.local` file. To adjust a variable's setting, you would open `hvenv.local` with a text editor of your choice and modify (or add) the appropriate line.

A typical `export` directive would appear as follows:

```
export SCRIPTS_TIME_OUT=200
```

When RMS starts, it reads the values of environment variables from `hvenv` and `hvenv.local` and initializes the `ENV` and `ENVL` objects respectively. No further reference is made to these two configuration files while RMS is running. Therefore, any changes you make to `hvenv.local` will not take effect until the next time RMS starts up.

Values in the `ENVL` (local) object override values in the `ENV` (global) object. If a global variable setting appears in the `hvenv.local` file, it will override the corresponding setting in the `hvenv` file. However, if you adjust a global variable in the `hvenv.local` file on one node, you must make the same adjustment to `hvenv.local` on every other node in the cluster. Global variable settings must agree clusterwide.

While RMS is running, you can display the environment variables with the `hvdisp` command, which does not require root privilege:

- `hvdisp ENV`
- `hvdisp ENVL`

13.2 Global environment variables



Global variable settings (`ENV`) are included in the configurations checksum that is common to the cluster. The checksum is verified on each node during startup of the base monitor. RMS will fail to start if it detects a checksum difference between the values on any two nodes.



The default values of the environment variables are found in `<RELIANT_PATH>/bin/hvenv`. They can be redefined in the `hvenv.local` configuration file.

The following list describes the global environment variables for RMS:

`HV_AUTOSTARTUP_IGNORE`

Possible values: List of RMS cluster nodes. The list of RMS cluster nodes must be the names of the `SysNodes` as found in the RMS configuration file. The list of nodes cannot include the CF name.

Default: "" (empty)

List of cluster nodes that RMS ignores when it starts. This environment variable is not set by default. A user application will begin its automatic startup processing if the `AutoStartUp` attribute is set and when all cluster nodes defined in the user application have reported `Online`. If a cluster node appears in this list, automatic startup processing will begin even if this node has not yet reported the `Online` state.

Use this environment variable if one or more cluster nodes need to be taken out of the cluster for an extended period and RMS will continue to use the configuration file that specifies the removed cluster nodes. In this

case, specifying the unavailable cluster nodes in this environment variable ensures that all user applications are automatically brought online even if the unavailable cluster nodes do not report `Online`.



Caution

If the `HV_AUTOSTARTUP_IGNORE` environment variable is used, ensure that it is correctly defined on all cluster nodes and that it is always kept up-to-date. When a node is brought back into the cluster, remove it from this environment variable. If this does not occur, data loss could occur because RMS will ignore this node during the startup procedure and will not check whether the application is already running on the nodes specified in this list. It is the system administrator's responsibility to keep this list up-to-date if it is used.

HV_AUTOSTART_WAIT

Possible values: 0–MAXINT

Default: 60 (seconds)

Defines the period (in seconds) that RMS waits for cluster nodes to report `Online` when RMS is started. If this period expires and not all cluster nodes are online, a switchlog message indicates the cluster nodes that have not reported `Online` and why the user application(s) cannot be started automatically.



This attribute generates a warning message only. `AutoStartUp` will proceed even if the specified period has expired.

HV_CHECKSUM_INTERVAL

Possible values: 0–MAXINT

Default: 120 (seconds)

Interval in seconds for which the RMS base monitor waits for each `Online` node to verify that its checksum is the same as the local checksum.

If checksums are confirmed within this interval, then RMS on the local node continues its operations as usual. However, if a checksum from a remote node is not confirmed, or if it is confirmed to be different, then the local monitor shuts down if it has been started less than `HV_CHECKSUM_INTERVAL` seconds before.

Also, if a checksum from a remote node is not confirmed, or if the checksum is confirmed to be different, then the local monitor considers the remote node as `Offline` if that local monitor has been started more than `HV_CHECKSUM_INTERVAL` seconds before.

HV_COM_PORT

Possible values: 0–65535

Default: 8000

The communication port used by the RMS base monitor on all the nodes in the cluster.

HV_LOG_ACTION_THRESHOLD

Possible values: 0–100

Default: 98

Determines conditions under which `hologcontrol` cleans up RMS log files. If the percentage of used space on the file system containing `RELIANT_LOG_PATH` is greater than or equal to this threshold, all subdirectories below `RELIANT_LOG_PATH` will be removed. Furthermore, if `HV_LOG_ACTION` is set to `on` and all subdirectories have already been removed, the current log files will be removed too. See `HV_LOG_ACTION` for more information.

HV_LOG_WARN_THRESHOLD

Possible values: 0–100

Default: 95

Defines when `hologcontrol` warns the user about the volume of RMS log files. If the percentage of used space on the file system containing `RELIANT_LOG_PATH` is greater than or equal to this threshold value, `hologcontrol` issues a warning to the user. See also `HV_LOG_ACTION_THRESHOLD` above.

HV_LOH_INTERVAL

Possible values: 0–`MAXINT`

Default: 30

Specifies the minimum difference in seconds when comparing timestamps to determine the last online host (LOH) for `userApplication`. It is determined if the `OnlinePriority` attribute is set to 1.

If the difference between the LOH timestamp entries logged in the `userApplication` on two cluster nodes is less than the time specified by this attribute, RMS does not perform `AutoStartUp` and does not allow priority switches. Instead, it sends a message to the console and waits for operator intervention.

When adjusting this variable, the quality of the time synchronization in the cluster must be taken into account. The value must be larger than any possible random time difference between the cluster hosts.

HV_USE_ELM

Possible values: 0, 1

Default: 1

Specifies the heartbeat monitoring mode used by the RMS base monitor:

0—remote node and base monitor states are detected by periodically sending UDP heartbeat packets across the network. If no heartbeats are received from a remote node during an interval defined by `HV_CONNECT_TIMEOUT`, RMS marks the node as down and waits for a recovery period before taking further action.

1—combines the Enhanced Lock Manager (ELM) method and the UDP heartbeat method. This setting is valid only when CF is installed and configured. The ELM lock is taken and held by the local node until ELM reports a remote node down or remote base monitor down. In either of these cases, the remote node is immediately killed. Until ELM reports a change in a remote node's state, RMS also monitors the UDP heartbeat of each remote node as described above, but with a much longer recovery timeout.

Whether or not ELM is enabled, a remote node is killed if its UDP heartbeat is not received before its heartbeat recovery timeout expires. When CF is not present, ELM is disabled automatically, and the heartbeat recovery timeout defaults to 45 seconds. When CF is present, ELM is enabled by default, and the heartbeat recovery timeout defaults to 600 seconds; this avoids premature node kills when the remote node is slow to respond.

Only experts should disable ELM manually. When CF is present but ELM is disabled, the default 600 second heartbeat recovery timeout is too long for efficient detection of remote RMS or node outages. In this case, the recovery timeout on the local node must also be adjusted manually by starting RMS with the `'hvcn -h <timeout> -c <config_file>'` command.

Note that the recovery timeout should be set to the same value on every node in the cluster. When ELM is disabled, the recommended global value is 45 seconds.

RELIANT_LOG_LIFE

Possible values: Any number of days

Default: 7 (days)

Specifies the number of days that RMS logging information is retained. Every time RMS starts, the system creates a directory that is named on the basis of when RMS was last started, and which contains all constituent log files. All RMS log files are preserved in this manner. All sub directories under `RELIANT_LOG_PATH` whose update time (ctime) is older than the the number of days specified in this variable are deleted by a cron job.

RELIANT_LOG_PATH

Possible values: Any valid path

Default: `/var/opt/SMAWRrms/log`

Specifies the directory where all RMS and Wizard Tools log files are stored. The location to store the internal log will not be changed even if the value of this variable is changed.

RELIANT_PATH

Possible values: Any valid path

Default: `/opt/SMAW/SMAWRrms`

Specifies the root directory of the RMS directory hierarchy. Users do not normally need to change the default setting.

RELIANT_SHUT_MIN_WAIT

Possible values: 0–`MAXINT`

Default: `MAXINT` (seconds)

Defines the period (in seconds) until the `hvshut` command times out.

If the `hvshut` command is executed with any of the `-l`, `-s`, and `-a` options, RMS performs shutdown processing after it performs offline processing of the active applications.

Set the total time of the following items for `RELIANT_SHUT_MIN_WAIT`:

1. Maximum time required to complete offline processing of applications
2. Maximum time required to shut down BM (base monitor) (30 seconds)

Use the total value of the script timeout of all resources included in an application for the value of 1.

To check the script timeout value of each resource, execute the `hvdisp` command by using the resource name as the argument and see the setting value (in seconds) of the `ScriptTimeout` attribute of the resource with which the `OfflineScript` attribute is not blank.

If there are two or more applications, use the largest value of the total values of the script timeout for each application.



By increasing the value of `RELIANT_SHUT_MIN_WAIT`, the following effects occur if delay or hang-up in offline processing of applications occurs.

- It may take longer time than the setting value of `RELIANT_SHUT_MIN_WAIT` to shut down RMS or the operating system.
- It may take longer time than the setting value of `RELIANT_SHUT_MIN_WAIT` to automatically switch applications by shutting down RMS or the operating system.

If the value of `RELIANT_SHUT_MIN_WAIT` is too large, use the processing time in the case that it is expected to take the longest time until timeout of the assumed cases where offline processing of an application times out for the value of the item 1 above.

Note that if the value of `RELIANT_SHUT_MIN_WAIT` is too small, timeout of the `hvshut` command may frequently occur before offline processing of applications completes. Tune `RELIANT_SHUT_MIN_WAIT` carefully.



Caution

Resources may remain active without stopping because RMS terminates abnormally when the `hvshut` command times out. Under this situation, if you start RMS on a different node and forcibly activate the application, the resources may become active on several nodes at the same time, and data may be corrupted when a shared disk is controlled by the resources. For this reason, if the `hvshut` command times out, shut down the operating system of the node where RMS has terminated abnormally, or forcibly shut down the node to ensure that resources stop. Then, start RMS and applications.

13.3 Local environment variables

Local environment variable settings can vary from node to node. The following list describes the local environment variables for RMS:

HV_AUTOSTARTUP

Possible values: 0, 1

Default: 1 (normal processing of `AutoStartUp` attribute)

Controls the action of the `AutoStartUp` attribute for all `userApplication` objects on the local node. If set to 1 (the default value) the automatic startup of each `userApplication` is determined by its `AutoStartUp` attribute (see the "Attributes available to the user"). If set to 0, the `AutoStartUp` attribute is ignored and no automatic startup occurs. `HV_AUTOSTARTUP` can be set in the `Cluster Admin Tools` menu or by using the `hvsetenv` command; in either case, the change does not take effect until the next RMS startup.

HV_CONNECT_TIMEOUT

Possible values: 5–`MAXINT`

Default: 30 (seconds). Users do not normally need to change the default setting.

The maximum time (in seconds) that the heartbeat from a node is not received before the base monitor assumes the connection to that node has been lost and starts the UDP heartbeat recovery timer.

Input values less than 5 are converted internally to 5.

HV_LOG_ACTION

Possible values: on, off

Default: off

Determines whether the current log files in the `RELIANT_LOG_PATH` directory will be deleted when the percentage of used space on the file system containing `RELIANT_LOG_PATH` is greater than or equal to `HV_LOG_ACTION_THRESHOLD`. See `HV_LOG_ACTION_THRESHOLD` for more information.

HV_MAX_HVDISP_FILE_SIZE

Possible values: 0–`MAXINT`

Default: 20,000,000 (bytes)

Prevents the unlimited growth of the temporary file that RMS uses to supply `hvdisp` with configuration data and subsequent configuration and state changes. The value of this variable is the maximum size in bytes of the temporary file `<RELIANT_PATH>/locks/.rms.<process id of the hvdisp process>`.

HV_MAXPROC

Possible values: 0–99

Default: 30

Defines the maximum number of scripts that RMS can execute at one time. The default (30) is sufficient in most cases.

HV_MLOCKALL

Possible values: 0, 1

Default: 0

If set to 1, the base monitor process and any memory it allocates will be locked in memory. If set to 0 (the default), the base monitor may be swapped out.

HV_RCSTART

Possible values: 0, 1

Default: 1 (start RMS in the `rc` script)

Determines if RMS is started in the `rc` script. If set to 1 (the default value), RMS is started automatically at system boot time. If set to 0, RMS must be started manually. `HV_RCSTART` can be set in the Cluster Admin *Tools* menu or by using the `hvsetenv` command. (Prerequisite for `rc start`: `CONFIG.rms` exists and contains a valid entry.)

HV_REALTIME_PRIORITY

Possible values: 1–99

Default: 50

Defines the real time priority for the RMS base monitor and its detectors. Caution should be used when adjusting this variable. High settings can prevent other OS real-time processes from getting their processor time slice. Low settings can prevent the RMS base monitor from reacting to detector reports and from performing requests from command line utilities.

This variable is processed only on Solaris platforms. It has no effect on Linux platforms.

HV_SCRIPTS_DEBUG

Possible values: 0, 1

Default: 0

Controls debugging output from RMS scripts. If this variable is set to 1, it overrides the setting for scripts that are generated and managed by the Wizard Tools, causing them to write detailed runtime information about the commands that are executed to the RMS `switchlog` file. The type of information logged may vary according to the script. This setting applies only to those scripts provided with PRIMECLUSTER products. To disable script debug message logging, delete the `HV_SCRIPTS_DEBUG` entry or set `HV_SCRIPTS_DEBUG=0` in `hvenv.local`.

Note: when this variable appears in `hvenv.local`, RMS adds it to the script environment but otherwise makes no attempt to process it. Therefore, it is not reported in the Cluster Admin GUI or in 'hvdisp ENVL' output.

HV_SYSLOG_USE

Possible values: 0, 1

Default: 1 (in `hvenv`)

Controls output to the system log from the RMS base monitor. RMS always records RMS ERROR, FATAL ERROR, WARNING, and NOTICE messages in the RMS `switchlog` file. By default, these messages are duplicated in the system log file `/var/adm/messages` (Solaris) or `/var/log/messages` (Linux). To disable RMS messages in the system log, set `HV_SYSLOG_USE=0` in `hvenv.local`.

**Note**

It is recommended for Linux to change this variable to 0. If this variable is set to 1 on Linux, under a high load on the system, the system log output will hang due to the performance limit of the system log processing by OS. In this case, RMS may not respond to any heartbeat from other node.

RELIANT_HOSTNAME

Possible values: valid name

Default: `<nodename>RMS`

The name of the local node in the RMS cluster. The default value of this variable is the node name with an RMS suffix (for example: `shasta1RMS`), as generated by the following command:

```
export RELIANT_HOSTNAME=`cftool -l 2>/dev/null | \
```



```
tail -1 | cut -f1 -d" "`RMS
```

If this preset value is not suitable, it must be modified accordingly on all the nodes in the cluster.

The specified cluster node name must correspond to the `SysNode` name in the `<configname>.us` configuration file. The node name determines the IP address that RMS uses for establishing contact with this node.

RELIANT_INITSCRIPT

Possible values: any executable

Default: `<RELIANT_PATH>/bin/InitScript`

Specifies an initialization script to be run by RMS when the system is started. This script is run before any other processes are activated. It is a global script that is run once on every cluster node on which it is defined.

RELIANT_STARTUP_PATH

Possible values: any valid path

Default: `<RELIANT_PATH>/build`

Defines where RMS searches at start time for the configuration files.

SCRIPTS_TIME_OUT

Possible values: 0–`MAXINT`

Default: 300 (seconds)

Specifies the global period (in seconds) within which all RMS scripts must be terminated. If a specific script cannot be terminated within the defined period, it is assumed to have failed and RMS begins appropriate processing for a script failure.

If this value is too low, error conditions will be produced unnecessarily, and it may not be possible for the applications to go online or offline. An excessively high value is unsuitable because RMS will wait for this period to expire before assuming that the script has failed.

In case the global setting is not appropriate for all objects monitored by RMS, this global value can be overridden by an object-specific setting of the `ScriptTimeout` attribute.

13.4 Script execution environment variables

The variables in this section are set by the RMS base monitor when it executes an object's script. These exist only in the script's environment and only for the duration of the script execution. Since these variables are explicitly set, they have no default values.

HV_APPLICATION

Possible values: any userApplication name

Name of the userApplication object at the top of the sub-tree that contains the current object.

HV_AUTORECOVER

Possible values: 0, 1

If set to 1, the script was initiated due to an AutoRecover attempt.

HV_FORCED_REQUEST

Possible values: 0, 1

If set to 1, the script is currently processing a forced request.

HV_LAST_DET_REPORT

Possible values: one of Online, Offline, Standby, Faulted

Last detector report for the current object.

HV_OFFLINE_REASON

Possible values: one of DEACT, SWITCH, FAULT, STOP, SHUT

Reason for a resource to become Offline is set:

SWITCH: switch request from userApplication ('hvswitch').

STOP: stop request from userApplication ('hvutil -f', 'hvutil -c').

FAULT: resource failure

DEACT: deact request from userApplication ('hvutil -d')

SHUT: stop request from RMS ('hvshut').

HV_NODENAME

Possible values: any object name

Name of current object.

HV_REQUESTING_CONTROLLER

Possible values: controller name and node name

If non-empty, contains the name of the controller and node that initiated the request for the current script execution.

HV_SCALABLE_CONTROLLER

Possible values: scalable controller name

The name of the scalable controller that controls this application.

HV_SCALABLE_INFO

Possible values: scalable applications list

Contains the list of all scalable applications everywhere in the cluster.

HV_SCRIPT_TYPE

Possible values: any of the following script types: PreCheck, PreOnline, Online, PostOnline, PreOffline, Offline, PostOffline, OfflineDone, Fault

NODE_SCRIPTS_TIME_OUT

Possible values: 0–MAXINT

Timeout value for the current object and script type.



HV_REQUESTING_CONTROLLER, HV_SCALABLE_CONTROLLER and HV_SCALABLE_INFO can be used only by the Online/Offline processing of the application controlled with the controller object.

14 Appendix—Troubleshooting

This chapter discusses some PRIMECLUSTER facilities for debugging RMS configurations from both the command line interface (CLI) and from the Cluster Admin graphical user interface (GUI). This chapter provides details on log files, their location, how to turn on logging levels, how to view logs from the GUI, and how to view log files from CLI.

Chapter contents:

- "Overview" summarizes the troubleshooting process.
- "Debug and error messages" describes RMS debug and error messages.
- "Log file categories" classifies the RMS log files.
- "Managing base monitor log output" describes how to manage logs produced by the RMS base monitor and its components.
- "Managing application log output" describes how to manage logs produced by the standard RMS Wizard Tools application and resource components.
- "Managing detector log output" describes how to manage logs produced by the standard RMS Wizard Tools resource detectors.
- "Interpreting RMS log messages" describes the formats of messages produced by the components of the RMS base monitor, and the applications and resources configured by the RMS Wizard Tools.
- "Viewing RMS log messages" describes the log viewer facility available in the Cluster Admin GUI.
- "RMS log file cleanup" describes how to control the volume of RMS log files either automatically or manually.
- "RMS troubleshooting" supplies solutions to problems that could occur while using RMS.
- "Collecting information for advanced troubleshooting" describes dump commands for consultants and developers.

14.1 Overview

The RMS troubleshooting process usually begins after you observe an error condition or unexpected state change in Cluster Admin in one of the following areas:


- Clusterwide table
- RMS tree
- Graph


The clusterwide table contains summary information and is a good place to start looking for error conditions. For additional details, you can look at the RMS tree or the graph. Depending on whether you need to look at the switchlogs or application logs, you can then use the log viewer facility to view the log files.

The log viewer has search facilities based on the following:

- Keywords
- Severity
- Non-zero exit codes

Search for causes of errors using the keywords and the date range fields. For emergency, alert, and critical conditions, you can do a search based on severity. For proactive troubleshooting, you can perform a search based on severity for the error, warning, notice, and info severity codes.

 It is recommended that you periodically use the log viewer and check the log files based on the severity levels to avoid serious problems. If you cannot diagnose the cause of a problem, look at the log viewer from two or more nodes in the cluster.

 Refer to the "RMS troubleshooting" for an explanation on corrective action.

Resolve error conditions as follows:

1. Use the Cluster Admin GUI.
2. View the log files if needed.
3. Change log levels to get more details.

4. If you cannot resolve an error condition with the GUI, you can use the command line interface. Use standard UNIX commands.
5. If a problem persists, check if it is a non-RMS issue and refer to the appropriate manual.
6. Check for system-related issues like operating system, hardware, or network errors.
7. Contact field engineers if you cannot resolve the issue.

14.2 Debug and error messages

RMS writes debug and error messages to log files when its components (such as the base monitor or detectors) operate. The default setting is for RMS to store these files in the `/var/opt/SMARrms/log` directory. Users can change the directory with the `RELIANT_LOG_PATH` environment variable, which is set in the `hvenv.local` file.

When RMS starts, logging begins. The default setting is for the base monitor to write all error messages to its log file or to `stderr`. Normally, you do not need to change the default setting because the default options allow for very detailed control of debug output.

If required, you can use the base monitor to record every state and message of any node. However, in most cases, the information requires a detailed knowledge of internal RMS operation to interpret the debug output, which can only be evaluated by field engineers.

For the administrator of an RMS cluster, evaluating the `switchlog` file is normally sufficient. This file records all important RMS actions; for example, incoming switch requests or faults that occur in nodes or resources.



There are also configuration-specific log files in the log directory. It is recommended that administrators evaluate these if necessary. The name of each log file (normally `<userApplication>.log`) depends on the configuration that was set up using the RMS Wizard Tools. Refer to the chapter "Managing application log output" for further information.

The `bmlog` file can also be useful for problem solving.

14.3 Log file categories

Table 10 identifies and explains the RMS log files contained in `/var/opt/SMAWRrms/Log`.

Module	File Name	Contents
Everything (base monitor, generic detector)	<code>switchlog</code>	Operational events, such as resource switches or bugs. Normally, <code>switchlog</code> is the only log file users need to examine.
generic detector	<code><detname>g<n>.log</code>	All messages and job assignments received by the detector. Also contains resource state change information and all error messages. <i>detname</i> is the name of the detector (<code>hvdet_*</code>) in the <code><RELIANT_PATH>/bin</code> directory.
base monitor	<code>abortstartlog</code>	This file contains records about <code>bm</code> exit conditions to assist field engineers in determining why RMS failed to start. This file is generated and the following console message appears when startup failures occur: FATAL ERROR: RMS has failed to start! Further details about the problem may appear in the <code>switchlog</code> .

Table 10: Log files

Module	File Name	Contents
base monitor	bmlog	<p>General RMS error and message logging information ranges from simple message reporting to more complete information. The error log level determines the contents of this file, which is specified when the base monitor is started. See "Managing base monitor log output" for more information.</p> <p>This file includes all messages received by the base monitor at runtime. It also contains information generated by <code>hvdump</code> (see "Using the <code>hvdump</code> command (RMS)").</p> <p>Should be used by experts only, since turning on log level flags consumes a great deal of disk space. By default, RMS places no messages in <code>bmlog</code>.</p>
base monitor	tracelog	<p>Records all messages between objects and all modification instructions. By default, RMS places no messages in <code>tracelog</code>.</p>
Object's detector and script related to applications	<application_name>.log	<p>All messages related to applications are recorded. The output of all the scripts which are executed by the application is also written to the log file.</p>

Table 10: Log files

14.4 Managing base monitor log output

At runtime, the RMS base monitor can generate log messages in the `RELIANT_LOG_PATH/switchlog` file and in the system log file. This section describes how you can control the **log level** (the amount of detail) in the log files. Each log level reports an internal function that is intended for expert use.

i Executing RMS with several active log levels will affect system performance. This feature should be enabled for testing or debugging purposes only.

14.4.1 Managing base monitor log levels

By default, base monitor logging is not running when RMS starts up. You can set the initial log level by specifying `hvcm` with the `-l` option:

```
hvcm -l <level> -c <configuration_file> ...
```

If RMS is already running, you can use the following commands to manage the base monitor log level:

```
hvutil -l display
hvutil -l off
hvutil -l <level>
```

The first `hvutil` command displays the current base monitor log level setting, and the second `hvutil` command turns off base monitor logging.

Specifying '`-l <level>`' with `hvcm` or `hvutil` activates base monitor logging, where `<level>` is one or more numbers in one of the following formats:

- A single number.
- A comma-delimited or space-delimited list. If the list is space-delimited, enclose the list in quotes. Examples: `2,4,5,7` "`2 4 5 7`"
- A single hyphen-separated range in the form `n1-n2`. This includes all log levels from `n1` up to and including `n2`. Specifying `-n2` is the same as `1-n2`, and specifying `n1-` includes all log levels above `n1`. The `n1` value must be greater than or equal to 1. Examples: `2-7` `-7` `2-`



- Specifying a log level of 0 (zero) activates **all log levels**. You must specify the special level `off` to deactivate logging.
- If RMS is being operated continuously for one day or more days when the base monitor logging is activated, to avoid the RMS logs taking up the disk space, configure the `cron` job setting to execute the `hvlogclean` command once a day. According to the setting, the log files are saved for each day. The older log files beyond the days specified by `RELIANT_LOG_LIFE` are deleted automatically. For details on the `hvlogclean` command, see the section "hvlogclean".

The current log level remains in effect until another `‘hvutil -l <level>’` is issued, a `‘hvutil -l off’` is issued, or until RMS is shut down.

Table 11 lists the valid base monitor log levels.

Log Level	Meaning
0	Turn on all log levels
1	Unused
2	Turn on detector tracing
3	hvdisk level
4	Turn on <code>mskx</code> tracing (stack tracing of the base monitor)
5	Error or warning message
6	Heartbeats and communication
7	Base monitor level
8	Generic controller message
9	Administrative command message
10	Basic-type level
11	Dynamic reconfiguration contracting level
12	Shutdown debug level
13	Token level
14	Detector message

Table 11: Base monitor log levels

Log Level	Meaning
15	Local queue level
16	Local queue level
17	Script level
18	userApplication contract level
19	Temporary debug traces
20	SysNode traces
21	Message level
22	bm tracelog
23	Unused

Table 11: Base monitor log levels

14.4.2 Controlling base monitor output to the system log

By default, the RMS base monitor writes the same messages to both the `switchlog` file and the system log file. This behavior is controlled by the `HV_SYSLOG_USE` environment variable.

The default setting in `hvenv` is `HV_SYSLOG_USE=1`. This sends all RMS NOTICE, WARNING, ERROR, and FATAL ERROR messages to the system log and `switchlog`.

To suppress RMS messages in the system log, edit the `hvenv.local` file and set `HV_SYSLOG_USE=0`. You will have to restart RMS for the change to take effect.

Note: For Log3 RMS messages, the component number is 1080023.

14.5 Managing application log output

Runtime components provided by the RMS Wizard Tools write log messages to files in the same log directory as the RMS base monitor, which is defined in the `RELIANT_LOG_PATH` environment variable. Messages from these components can be broken down into two categories:

- Messages from resource detectors
- All other messages

Detector logging is described in "Managing detector log output". The following discussion applies to all other log messages.

The runtime components provided by the configuration tools log everything at an application level. In addition to the normal user-oriented messages, they can also generate debug messages with various levels of detail. The user and debug messages may be written to the following files in *RELIANT_LOG_PATH*:

- `switchlog`—The standard subapplications supplied by the RMS Wizard Tools record detector reports related to fault and offline transitions into the `switchlog` file.
- `<application_name>.log`—The application-specific log file records all messages associated with that application. The output from any scripts run by the application also go into its log file. The file is created when either offline or online processing for the application begins.
- `hvdet_<xxx>.g<n>.log`—These are detector log files which record all relevant information regarding the resources they are monitoring, such as all state transitions.

When debugging application problems, the `switchlog` file, the application-specific log file, and any appropriate detector log files may all need to be viewed and interpreted.

14.5.1 Controlling debug messages from standard scripts

You can control debug output from the standard RMS Wizard Tools scripts by setting the environment variable `HV_SCRIPTS_DEBUG` in the `hvenv.local` file:

- To turn on debug output, create the following entry:

```
export HV_SCRIPTS_DEBUG=1
```
- To turn off debug output, either remove the `HV_SCRIPTS_DEBUG` entry, comment it out, or set the value to 0.

14.6 Managing detector log output

Each instance of an RMS detector writes information to both the `switchlog` file and to its own dedicated log file, which has a name of the form

```
RELIANT_LOG_PATH/<detector_name>.g<n>log
```

where *<detector_name>* is the name of the detector and *<n>* is the instance number of the detector. There is one such file for every running instance of a detector. Note that the RMS Wizard Tools detectors are named `hvdet_<xxx>`, where *<xxx>* denotes the resource type.

For instance, the log file for the first (and only) `hvdet_system` detector is `<RELIANT_LOG_PATH>/hvdet_system.g1log`.

Resource state changes are recorded in both the `switchlog` file and the detector's dedicated log file. All other detector messages are recorded only in the dedicated log file.

14.6.1 Debug messages and log levels

Each detector instance maintains an internal 10 KB memory buffer that it uses to maintain debug messages. These are written to the dedicated log file when a debug event (an unexpected resource status report) occurs. The buffer is circular, so when a new message would cause the buffer to overflow, it will instead overwrite one or more of the oldest stored messages. This continues until a debug event causes the entire buffer to be written to the log file, at which time the buffer is reset. Therefore, only the latest messages will be logged. Some older data may be lost, depending on when a debug event occurs.

When the buffer contents are flushed to the log file, the messages are written in chronological order. It is important to note that the latest message in the buffer was generated by the debug event, but earlier messages in the buffer may or may not be related. Therefore, when you examine the detector log, you may find a series of benign messages, followed by a debug event message, followed by more benign messages, followed by another debug event message, etc. This means you must check the timestamp and text of every detector debug message to see if it pertains to the problem you're trying to resolve.

Every message that the detector can generate has an assigned detail level, where higher detail levels produce more debugging information. At runtime, only those messages with a detail level less than or equal to the detector's current log level will be added to the internal buffer. The default value of each detector log level is set to 1.

i The amount of information produced by a specific log level depends on the detector resource type. A high log level for one detector may produce less output than a low log level for another detector.

14.6.2 Setting the detector log level

You can set individual log levels for each resource detector from the command line and the RMS Wizard Tools.

i To enable detector logging, the RMS base monitor log level **must include level 2** (detector tracing), regardless of the individual detector log level settings. By default, the base monitor log level is `off`, so detector logging is suppressed. See "Managing base monitor log output".

14.6.2.1 Setting the detector log level with `hvutil -L`

If RMS is already running, you can use the following commands to manage the detector log level:

```
hvutil -L display <resource>
```

```
hvutil -L <level> <resource>
```

The first `hvutil` command displays the current detector log level for the indicated resource, and the second form of the command sets the log level:

- `<level>` can be either 0 (zero) to turn off logging for the resource, or a positive number to turn on logging at that level.
- `<resource>` is the name of the resource monitored by the detector. This must be a generic resource, *i.e.*, it must be an object of type `gResource`. You can find these in the `hvdisk -a` output.

The current log level remains in effect until another `‘hvutil -L’` command changes the log level, `‘hvutil -l off’` is issued, or until RMS is shut down.

14.6.2.2 Setting the log level in the Wizard Tools

The valid range of log levels is 1 to 9, and the default value is 1. The log level can be modified from the `hvw` command as follows:

1. Select the *Configuration-Edit-Global-Settings* menu.
2. Choose the *DetectorDetails* sub-menu. The screen in Figure 160 appears.

```

Detector Details:
 1) HELP                               12) ReactionTime0fhvdet_nfs=10
 2) RETURN                             13) ReactionTime0fhvdet_rcvm=33
 3) DEFAULTVALUES                     14) ReactionTime0fhvdet_read=10
 4) ReactionTime0fhvdet_ckhost=10     15) ReactionTime0fhvdet_srdf=31
 5) ReactionTime0fhvdet_ddm=19        16) ReactionTime0fhvdet_stopclnt=10
 6) ReactionTime0fhvdet_execbin=10    17) ReactionTime0fhvdet_system=10
 7) ReactionTime0fhvdet_glbassrt=10   18) ReactionTime0fhvdet_vxvm=30
 8) ReactionTime0fhvdet_gmount=10     19) ReactionTime0fForeignDetectors=30
 9) ReactionTime0fhvdet_icmp=10       20) MemoryLogLevel=1
10) ReactionTime0fhvdet_locassrt=10   21) DynamicDetectorLogging=0
11) ReactionTime0fhvdet_lvm=18
Set details for a detector:

```

Figure 160: Wizard Tools Detector Details menu

3. Select *MemoryLogLevel* and change the value to the desired level. The log level applies to all detectors.

Runtime adjustment with the Wizard Tools

It is possible to turn debug reporting on or off dynamically from the RMS Wizard detectors by using the `hvw` command as follows:

1. Select *Configuration-Edit-Global-Settings*.
2. Choose the *DetectorDetails* sub-menu.
3. Select the *DynamicDetectorLogging* menu item.

The default value is 0, which turns off debugging. Values in the range 1 to 9 turn on debugging, and higher numbers produce more information. The volume of generated log messages varies according to the detector, so a high value for one detector may produce fewer messages than a low value for another. Changes to this value do not take effect until the next time the configuration is activated.

Using `hvw` to turn on dynamic logging actually creates the file `<RELIANT_PATH>/etc/wizardloglevel`, which simply contains the desired debug level as a single-digit ASCII number. You can bypass the `hvw` command and create (or delete) the `wizardloglevel` file manually, as long as you observe the following rules:

- If the file does not exist, or if the file contains the number 0 (zero), then debugging is turned off.
- If the file exists but is empty, a value of 3 is assumed.
- If the file contains a value in the range 1 to 9, debugging is turned on.



Turning on debugging should only be done when problems occur. Once the problems are resolved, debugging should be turned off to avoid filling the file system with extraneous information.

14.7 Interpreting RMS log messages

This section describes the format of messages from various RMS components.

14.7.1 switchlog message format

The `RELIANT_LOG_PATH/switchlog` file records RMS events relevant to the user, such as switch requests and fault indications. It contains the following five message types:

1. Informational messages (notices)
2. Warning messages
3. Error messages
4. Fatal error messages
5. Output from scripts run by RMS

Notices, warnings, errors, and fatal errors

The first four categories of messages all appear in this format:

timestamp: (err_code, err_number): msg_type: msg_text: msg_end

There is a colon followed by a space (:) between each field of the message. The fields are as follows:

1. *timestamp* has the format
`yyyy-mm-dd hh:mm:ss.xxx`
2. (*err_code*, *err_number*) is a two or three letter code denoting the internal component that generated the message, followed by the unique number of the message for that component.
3. *msg_type* is one of the following:
 - NOTICE
 - WARNING
 - ERROR
 - FATAL ERROR
4. *msg_text* is any text generated by the RMS product. This consists of one or more lines of text, each followed by a newline character.
5. *msg_end* is a series four equal signs (====) followed by a newline character. With the terminal newline of the preceding *msg_text* field and the intervening colon-space field delimiter, this appears as a colon, a space, and four equal signs on a separate line in the log file. This uniquely identifies the end of each message, regardless of the number of lines.

For listings of all RMS `ERROR` and `FATAL ERROR` messages (without the leading *timestamp* and trailing *msg_end* fields), see "PRIMECLUSTER Messages."

Script output

The fifth category of messages, script output, follows no specific format. These messages are simply the redirected standard output and standard error streams generated by scripts defined in the RMS configuration. They generally have neither a leading *timestamp* field nor a trailing *msg_end* field.

Here is an example of a short sequence of script messages:

```
Starting Reliant Monitor Services now
loadcount: 0
loadcount lt 3
```

14.7.2 Application log message format

Applications and resources configured by the RMS Wizard Tools produce log messages in the following format:

```
resource_name: state: timestamp: msg_type: msg_text: msg_end
```

There is a colon followed by a space (:) between each field of the message. The fields are as follows:

1. *resource_name* is the name of the particular resource node in the RMS graph whose script is running. This field may be empty if no resource is associated with the message.
2. *state* is an indication of the type of action that is being performed, and is the value as set by RMS in the environment variable `HV_SCRIPT_TYPE`. The field typically contains the values `online` or `offline`. The RMS configuration tools also set the field with the value `PreCheck`, when a `PreCheck` script is being run. This field will be empty for messages of type `DEBUG` being printed.
3. *timestamp* contains the date and time when the message was generated. It appears in the format `yyyy-mm-dd hh:mm:ss.xxx`.
4. *msg_type* is one of the following:
 - `DEBUG`
 - `NOTICE`
 - `WARNING`
 - `ERROR`
 - `FATAL ERROR`
5. *msg_text* contains the text generated by the RMS Wizard product. This text may consist of more than one line, *i.e.*, it may have **embedded** newline characters. In general, it is not **terminated** with a newline character.
6. *msg_end* is a series of four equal signs (====). Since *msg_text* is not usually terminated with a newline, *msg_end* appears on the same line.

14.7.3 Program log message format

RMS consists of a number of individual programs that contribute to the overall high availability management. Each of these programs can generate its own trace and error messages in a dedicated log file, which can be found at

```
RELIANT_LOG_PATH/<program_name>log
```

For example, the RMS base monitor program is named `bm`, and its dedicated log file is `RELIANT_LOG_PATH/bmlog`.

Program trace and error messages are intended for internal troubleshooting purposes only. The format is described here in case you are asked to provide the information to a PRIMECLUSTER consultant.

Trace messages have the following prefix:

```
timestamp: file: line:
```

Error messages have the following prefix:

```
timestamp: file: line: ERROR
```

Here is an example of a short sequence of trace messages:

```
2005-12-17 14:42:46.256: det_generic.C: 756: Return from  
GdCheck  
2005-12-17 14:42:46.256: det_generic.C: 773: Call to  
GdGetAttribute  
2005-12-17 14:42:46.257: det_generic.C: 775: Return from  
GdGetAttribute  
2005-12-17 14:42:46.257: det_generic.C: 790: reporting state  
to bm
```

14.8 RMS log file cleanup

RMS provides two methods for housekeeping of its log files:

1. `hvtlogclean` deletes or backs up log files based on elapsed time
2. `hvtlogcontrol` issues warnings or deletes files based on the available space in the file system

Both of these are automatically installed and configured with default settings as part of the RMS installation. This section outlines their functions and lists their control settings.

14.8.1 hvlogclean

The `hvlogclean` utility saves the current log files into a backup subdirectory of `RELIANT_LOG_PATH` whose name is the time RMS was last started. If invoked with the `-d` option, it will delete the current log files rather than copy them. In either case, `hvlogclean` creates a clean set of log files even while RMS is running.

`hvlogclean` also purges old files in the backup subdirectory. It determines whether or not a backup file is “old” by checking the value of the `RELIANT_LOG_LIFE` environment variable. If the backup file is older (in days) than the number in the variable, the file will be deleted. See `RELIANT_LOG_LIFE` for more information.

14.8.2 hvlogcontrol

The `hvlogcontrol` utility prevents log files from consuming all the free space on the file system containing `RELIANT_LOG_PATH`. According to the limits set in the configuration, `hvlogcontrol` can warn the user, delete only the subdirectories in `RELIANT_LOG_PATH`, or delete all RMS log files. See the descriptions of the following RMS environment variables for more information:

- `HV_LOG_ACTION_THRESHOLD` (global)
- `HV_LOG_WARN_THRESHOLD` (global)
- `HV_LOG_ACTION` (local)

Changing these controls requires editing the `hvenv.local` file and then restarting RMS. See "Setting environment variables".



`hvlogcontrol` is invoked periodically from the `crontab` file. It has no manual page.

14.8.3 Logging of background scripts

A Command Line subapplication can invoke a script that executes another command or script in the background. If the background process writes to `stdout` or `stderr`, that output will be directed to a log file that remains open

until the process ends normally or is terminated. This will cause the log file to be handled differently when RMS log files are cleaned via the `cron` job or `hvllogclean`.

When `hvllogclean` cleans up old log files, it moves them into a backup directory named according to the time RMS was last started. The open log file of the background process will be moved there, too. Because the background process continues to write into the open file, future output for the process will therefore be found in the backup directory, and not in `/var/opt/reliant/log` along with the rest of the RMS processes.

If the log file of the background process is removed while the process is still running (for example, either manually or by the `'hvllogclean -d'` command), the log file will no longer be visible. However, the file will still exist, and the background process will continue to write into the now-invisible file until the process ends or is stopped.

14.9 RMS troubleshooting

When problems occur, RMS prints out meaningful error messages that will assist you in troubleshooting the cause. If no message is available, the following information may help you diagnose and correct some unusual problems:


- RMS dies immediately after being started.

At startup, the RMS base monitor exchanges its configuration checksum with the other base monitors on remote nodes. If the checksum of the starting base monitor matches the checksums from the remote nodes, the startup process continues. If the checksums do not match, then the RMS base monitor shuts down if all of the following conditions are true:

1. The base monitor has encountered a different checksum from a remote monitor within the initial startup period (see `"HV_CHECKSUM_INTERVAL"`).
2. There are no applications on this node that are online, waiting, busy, or locked.
3. There are no online remote base monitors encountered by this base monitor.

Otherwise, the base monitor keeps running, but all remote monitors whose checksums do not match the local configuration checksum are considered to be offline. Therefore, no message exchange is possible with these monitors, and no automatic or manual switchover will be possible between the local monitor and these remote monitors.

When different checksums are encountered, certain messages are placed in the switchlog explaining the situation.

 Configuration checksum differences often result when global environment variables are changed manually but inconsistently on different nodes.

Action:

To verify that a configuration checksum difference is not the cause of the problem, ensure that all the nodes have been updated with the proper configuration by using the following procedure:

1. Stop RMS everywhere in the cluster. This may require an explicit `hvshut` command on any nodes that are not communicating with the others.
2. Determine which configuration to run. Use `'hvdisp -a'` or `'hvdisp -T SysNode'` on each node to verify the name of the configuration file. (The `hvdisp` command does not require root privilege.)

A configuration may have the same name but different contents on two or more nodes if one of the following has occurred:

- The most recent activation did not distribute the configuration to all the nodes in the cluster. For instance, the user may have specified *Ignore Down Nodes* when activating with Wizard Tools.
 - If the Wizards Tools were used on more than one node in the cluster, there may be more than one version of the configuration with the same name.
3. Activate the correct configuration with the same tool (Wizard Tools) that was used to create it. For the correct procedure, see the section "Activating a configuration".

Alternatively, redistribute the existing `<configname>.us` file with one of the following methods:

- In the RMS Wizard Tools, use *Configuration Push*.

Make sure the activation is successful so that all the nodes are updated.

4. Start RMS on the entire cluster. All the nodes should now be running the same configuration.

- RMS hangs after startup (processes are running, but `hvdisk` hangs)

This problem might occur if the local node is in the CF state `LEFTCLUSTER` from the point of view of one or more of the other nodes in the cluster.

Action:

Verify the problem by using `'cftool -n'` on all cluster nodes to check for a possible `LEFTCLUSTER` state.

Use `'cftool -k'` to clear the `LEFTCLUSTER` state. RMS will continue to run as soon as the node has joined the cluster. No restart should be necessary.

- RMS loops (or even dies) shortly after being started.

This problem could occur if the CIP configuration file `/etc/cip.cf` contains entries for the netmask. These entries are useless (not evaluated by CIP). From the RMS point of view these entries cannot be distinguished from IP addresses, which have the same format, so RMS will invoke a `gethostbyaddr()`. This normally does no harm, but in some unusual cases the OS may become confused.

Action:

Verify the problem by checking if netmask entries are present in `/etc/cip.cf`.

Remove the netmask entries, and restart RMS.

- RMS detects a failure of another node, but the node is not killed.

This problem could occur if `SysNode` is in the `Wait` state.

Action:

Check the CF state by using `cftool -n`.

If the CF state is `LEFTCLUSTER`, manually stop the `LEFTCLUSTER` node, and then clear the `LEFTCLUSTER` state by using `cftool -k`.

Once CF state changes from `LEFTCLUSTER` to `DOWN`, execute "`hvdisk -T SysNode`" to see the state of all `SysNode` objects.

If there is any `SysNode` in `Wait` state, execute "`hvutil -u SysNode`" to clear the `Wait` state.



Caution

When using the `cftool -k` command and the `hvutil -u` command, you must manually stop the node in the `Wait` state before executing the commands. The execution of these commands may invoke the failover of the applications. Therefore, if you execute these commands without stopping the node in the `Wait` state, this may cause data corruption.

- The RMS base monitor detects a loss of detector heartbeat, but there is no indication as to the reason for the loss.

RMS automatically invokes a tool that provides diagnostic information for this event. The diagnostic tool performs the following actions:

- Invokes `truss(1)` on Solaris or `strace(1)` on Linux to trace the detector process
- Turns on full RMS and detector logging with the `-l0` (lowercase “L”, zero) option
- Gathers system and users times for the process

The `truss(1)/strace(1)` invocation and logging levels will be terminated after the number of seconds specified in the `ScriptTimeout` attribute. All information is stored in the `switchlog` file.

Note that user-specified operations such as detector tracing will continue on each node, even if it appears to have left the cluster. In extreme cases, turning on high detail trace levels may affect the performance of a node and contribute to delays in its base monitor heartbeat.

Action:

See the `switchlog` for the diagnostic information.

14.10 Collecting information for advanced troubleshooting

The procedures in this section provide advanced debugging information and are intended for use by consultants and developers.

14.10.1 Using the `hvdump` command (RMS)

The `hvdump` command collects RMS debugging information on the local node. A normal RMS installation creates the file

```
RELIANT_PATH/bin/hvdump
```

By default, *RELIANT_PATH* is `/opt/SMAW/SMAWRrms/`. Symbolic links to the same file are also created in `/usr/bin/hvdump` (which is usually in the default search path) and `/opt/SMAW/bin/hvdump`.

Invoking `hvdump` gathers log files and configuration data and places them in the *RELIANT_PATH* directory in the following archive:

```
<nodename>RMS.<timestamp>.debug_information.tar.<suf>
```

where the suffix *<suf>* is 'gz' on Linux platforms, and 'Z' on Solaris platforms.

The archive file is intended for use by RMS support specialists. The `hvdump(1M)` manual page presents a detailed list of the files and information included in the archive.

14.11 SNMP Notification of Resource Failure

If an error was detected, SNMP Trap can be sent to the server on which the SNMP manager is running.

By using this function, the SNMP manager can monitor the cluster system.

For details on how to set this function, see "6.10.1 Setting Contents of a Cluster Application" in "PRIMECLUSTER Installation and Administration Guide (Linux)."

For SNMP Trap, the version 2c is used and the following pieces of information are contained in SNMP Trap to be sent.

- OIDs corresponding to the resource types

- Message indicating the resource failure

To send SNMP Trap, the `snmptrap` command is used and the following operation is performed.

```
snmptrap -v 2c -c <Community name> <Destination host> \
.1.3.6.1.4.1.211.4.68.257 <OID> s <Message>
```

The following table shows OIDs set for SNMP Trap.

No	OID	Description
1	.1.3.6.1.4.1.211.4.68.257.1	This OID is set when a resource failure occurs in an application or middleware. It is set when a failure is detected in the Cmdline resource, procedure resource, or in middleware resource.
2	.1.3.6.1.4.1.211.4.68.257.2	This OID is set when a network failure occurs. It is set when a failure is detected in the GIs resource, or in the takeover network resource.
3	.1.3.6.1.4.1.211.4.68.257.3	This OID is set when a file system failure is detected. It is set when a failure is detected in the Fsystem resource.
4	.1.3.6.1.4.1.211.4.68.257.4	This OID is set when a disk failure is detected. It is set when a failure is detected in the Gds resource.

Table 12: OID set in SNMP Trap

The message which RMS outputs to switchlog when the resource failure occurs is contained in "Message indicating a resource failure." The message indicates what failure is detected in which resource.

Example of a message:

```
2014-07-28 17:07:09.254:(DET, 7): ERROR: FAULT REASON:
Resource <ManageProgram000_Cmd_APP1> transitioned to a
Faulted state due to the resource unexpectedly becoming
Offline.
```

You can also confirm the cause and action of the resource failure by searching the message in "PRIMECLUSTER Messages."

**Note**

- If an error occurs in the communication path between the cluster node and the host on which the SNMP manager is running, the notification of resource failure by SNMP Trap does not work.
- If the `userApplication` is switched over when the multiple resource failures are detected in one `userApplication` at the same time, only one of the resource failures will be notified. Check if other resource failures occur in the switchlog file after the cause of the notified resource failure has been removed on the cluster node.

15 Appendix—RMS command line interface

The primary interface for configuring RMS is the RMS Wizard Tools, and the primary interface for administering RMS is the Cluster Admin GUI. These user interfaces call the RMS command line interface (CLI), and, under certain conditions, you may find it useful to invoke the CLI directly.

The following section lists the RMS CLI commands available to administrators. Specific procedures using some of these commands are described in the "Controlling RMS operation". For a complete description of any command's usage, see the "Controlling RMS operation". For a list of all PRIMECLUSTER commands related to RMS, see "Appendix B Manual Pages" in "PRIME-CLUSTER Installation and Administration Guide."



With few exceptions, RMS CLI commands require root privilege. The exceptions are noted in the following list.

15.1 Available RMS CLI commands

hvassert

Tests an RMS resource for a specified resource state. It can be used in scripts when a resource must achieve a specified state before the script can issue the next command. Does not require root privilege.

hvcm

Starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the `hvcm` command.

The base monitor is the decision-making module of RMS. It controls the configuration and access to all RMS resources. If a resource fails, the base monitor analyzes the failure and initiates the appropriate action according to the specifications for the resource in the configuration file.

hvconfig

Either displays the current RMS configuration or sends the current configuration to an output file.

The output of the `hvconfig` command is equivalent to the running RMS configuration file, but does not include any comments that are in the original file. Also, the order in which the resources are listed in the output might vary from the actual configuration file.

hvdisp

Displays information about the current configuration for RMS resources. Does not require root privilege.

hvdisplayall

Display the resource information of all the nodes in RMS.

hvdump

Gets debugging information about RMS on the local node.

hvllogclean

Either saves old log files into a subdirectory whose name is the time RMS was last started, or, if invoked with the `-d` option, deletes old log files. In either case, `hvllogclean` creates a clean set of log files even while RMS is running.

hvreset

Reinitializes the graph of an RMS user application on one or more nodes in the configuration. Running scripts will be terminated, ongoing requests and contracts will be cleaned up, and information about previous failures will be purged. If the process is successful, the entire graph will be brought back into a consistent initial state, but an inconsistent state is also a possible result. Therefore, use this command for test purposes only, and never invoke it on a production cluster.

**Caution**

This command is intended for use by experts only.

hvsetenv

Provides an interface for changing the following RMS environment variables on the local node:

- `HV_RCSTART` controls the automatic startup of RMS.
- `HV_AUTOSTARTUP` controls the automatic startup of all applications.
-

For more information about these environment variables, see "Appendix—Environment variables".

hvshtut

Shuts down RMS on one or more nodes in the configuration. The base monitor on the local node sends a message to other online nodes indicating which node or nodes will be shut down.

hvswitch

Manually switches control of a `userApplication` or a `gResource` between `SysNodes` in the RMS configuration.

hvutil

Provides general administration interface to RMS. It performs various resource administration tasks, such as dynamically setting logging levels, sending a resource `Offline`, clearing faulted resources or hung cluster nodes in the `Wait` state, setting detector time periods, setting Maintenance Mode, and so forth.

**Caution**

Setting high logging levels in `hvutil` can cause disk overflow if enabled for too long. See the section "Managing base monitor log levels" for more information.

16 Appendix—Release information

This appendix provides information about the changed sections in this manual.

No	VL	Edition	Location	Description
1	4.5A00	December 2017	"Main configuration menu when RMS is not active"	Added the note about main configuration menu.
2	4.5A00	December 2017	"Stopping RMS"	Added the note when stopping RMS.
3	4.5A00	December 2017	"Overriding automatic application startup"	Changed the explanation when automatically starting the application.
4	4.5A00	December 2017	"Attributes available to the user"	Changed the explanations of the following attributes available to the user: - AutoStartUp - OnlinePriority - PartialCluster
5	4.5A00	December 2017	"Global environment variables"	Changed the explanation of HV_AUTOSTART_WAIT.
6	4.5A00	December 2017	"Appendix—Troubleshooting"	Deleted the explanation of "Configuration troubleshooting."

Glossary

Items in this glossary that apply to specific PRIMECLUSTER components are indicated with the following notation:

- (CF)—Cluster Foundation
- (RMS)—Reliant Monitor Services
- (SIS)—Scalable Internet Services

Some of these products may not be installed on your cluster. See your PRIME-CLUSTER sales representative for more information.

AC

See *Access Client*.

Access Client

GFS kernel module on each node that communicates with the Meta Data Server and provides simultaneous access to a shared file system.

administrative LAN

An optional private local area network (LAN) used for administrative commands to the nodes in the cluster. To provide an extra level of security, normal users do not have access to the administrative LAN. In PRIMECLUSTER configurations, the System Console and Cluster Console reside on the administrative LAN if one is present.

See also public LAN.

API

See Application Program Interface.

application (RMS)

In the RMS context, an application object is a special resource used to group other resources into a logical collection. Typically, it is used to represent a real-world application or application suite in a high-availability configuration.

Application Program Interface

A shared boundary between a service provider and the application that uses that service.

application template (RMS)

A predefined group of object definition value choices used by the Wizard Tools or the Wizard Kit to create object definitions for a specific type of application.

attribute (RMS)

The part of an object definition that specifies how the base monitor acts and reacts for a particular object type during normal operations.

automatic power control

This function is provided by the Enhanced Support Facility (ESF), and it automatically switches the server power on and off.

automatic switchover (RMS)

The procedure by which RMS automatically switches control of a userApplication over to another node after specified conditions are detected.

See also directed switchover (RMS), failover (RMS, SIS), switchover (RMS), symmetrical switchover (RMS).

availability

Availability describes the need of most enterprises to operate applications via the Internet 24 hours a day, 7 days a week. The relationship of the actual to the planned usage time determines the availability of a system.

base cluster foundation (CF)

This PRIMECLUSTER module resides on top of the basic OS and provides internal interfaces for the CF (Cluster Foundation) functions that the PRIMECLUSTER services use in the layer above.

See also Cluster Foundation (CF).

base monitor (RMS)

The RMS module that maintains the availability of resources. The base monitor is supported by daemons and detectors. Each node being monitored has its own copy of the base monitor.

Cache Fusion

The improved interprocess communication interface in Oracle 9i that allows logical disk blocks (buffers) to be cached in the local memory of each node. Thus, instead of having to flush a block to disk when an update is required, the block can be copied to another node by passing a message on the interconnect, thereby removing the physical I/O overhead.

CCBR (Solaris)

See Cluster Configuration Backup and Restore (Solaris).

CF

See Cluster Foundation (CF).

CF node name (CF)

The CF cluster node name, which is configured when a CF cluster is created.

child (RMS)

A resource defined in the configuration file that has at least one parent. A child can have multiple parents, and can either have children itself (making it also a parent) or no children (making it a leaf object).

See also resource (RMS), object (RMS), parent (RMS).

cluster

A set of computers that work together as a single computing source. Specifically, a cluster performs a distributed form of parallel computing.

See also RMS configuration (RMS).

Cluster Admin

A Java-based, OS-independent management tool for PRIMECLUSTER products such as CF, SIS, and RMS. Cluster Admin is available from the Web-Based Admin View interface.

See also Cluster Foundation (CF), Scalable Internet Services (SIS), Reliant Monitor Services (RMS), Web-Based Admin View.

Cluster Configuration Backup and Restore (Solaris)

CCBR provides a simple method to save the current PRIMECLUSTER configuration information of a cluster node. It also provides a method to restore the configuration information.

Cluster Foundation (CF)

The set of PRIMECLUSTER modules that provides basic clustering communication services.

See also base cluster foundation (CF).

cluster interconnect (CF)

The set of private network connections used exclusively for PRIME-CLUSTER communications.

Cluster Join Services (CF)

This PRIMECLUSTER module handles the forming of a new cluster and the addition of nodes.

configuration file (RMS)

In the RMS context, the single file that defines the monitored resources and establishes the interdependencies between them. The default name of this file is `config.us`.

console

See single console.

custom detector (RMS)

See detector (RMS).

custom type (RMS)

See generic type (RMS).

daemon

A continuous process that performs a specific function repeatedly.

database node (SIS)

Nodes that maintain the configuration, dynamic data, and statistics in a SIS configuration.

See also gateway node (SIS), service node (SIS), Scalable Internet Services (SIS).

detector (RMS)

A process that monitors the state of a specific object type and reports a change in the resource state to the RMS base monitor.

DHCP

Dynamic Host Control Protocol. A standard method of delivering information to a host at boot time. This is most often used to dynamically assign the host's IP address and netmask, but many other parameters are possible, including domain names, DNS servers, and time servers.

directed switchover (RMS)

The RMS procedure by which an administrator switches control of a `userApplication` over to another node.

See also automatic switchover (RMS), failover (RMS, SIS), switchover (RMS), symmetrical switchover (RMS).

DOWN (CF)

A node state that indicates that the node is unavailable (marked as down). A LEFTCLUSTER node must be marked as DOWN before it can rejoin a cluster.

See also UP (CF), LEFTCLUSTER (CF), node state (CF).

Enhanced Lock Manager (ELM) (CF)

A light weight, high performance, highly responsive lock manger, specifically designed for providing a high reliability heartbeat messaging mechanism for PRIMECLUSTER modules.

ENS (CF)

See Event Notification Services (CF).

environment variables

Variables or parameters that are defined globally.

error detection (RMS)

The process of detecting an error. For RMS, this includes initiating a log entry, sending a message to a log file, or making an appropriate recovery response.

Event Notification Services (CF)

This PRIMECLUSTER module provides an atomic-broadcast facility for events.

failover (RMS, SIS)

With SIS, this process switches a failed node to a backup node. With RMS, this process is known as switchover.

See also automatic switchover (RMS), directed switchover (RMS), switchover (RMS), symmetrical switchover (RMS).

Fast switching mode

One of the LAN duplexing modes presented by GLS. This mode uses a multiplexed LAN simultaneously to provide enhanced communication scalability between servers, and highspeed switchover if a LAN failure occurs.

gateway node (SIS)

Gateway nodes have an external network interface. All incoming packets are received by this node and forwarded to the selected service node, depending on the scheduling algorithm for the service.

See also service node (SIS), database node (SIS), Scalable Internet Services (SIS).

generic type (RMS)

An object type which has generic properties. A generic type is used to customize RMS for monitoring resources that cannot be assigned to one of the supplied object types.

See also object type (RMS).

Global Disk Services

This optional product provides volume management that improves the availability and manageability of information stored on the disk unit of the Storage Area Network (SAN).

Global File Services

This optional product provides direct, simultaneous accessing of the file system on the shared storage unit from two or more nodes within a cluster.

Global Link Services

This PRIMECLUSTER optional module provides network high availability solutions by multiplying a network route.

generic type (RMS)

An object type which has generic properties. A generic type is used to customize RMS for monitoring resources that cannot be assigned to one of the supplied object types.

See also object type (RMS).

graph (RMS)

See system graph (RMS).

graphical user interface

A computer interface with windows, icons, toolbars, and pull-down menus that is designed to be simpler to use than the command-line interface.

GUI

See graphical user interface.

high availability

A system design philosophy in which redundant resources are used to avoid single points of failure.

See also Reliant Monitor Services (RMS).

Intelligent Platform Management Interface

A firmware and hardware specification that provides common interfaces for monitoring and managing computers. IPMI operates through an onboard Baseboard Management Controller (BMC) on the target machine to provide OS-independent remote management functions, whether or not the target machine is powered on.

interconnect (CF)

See cluster interconnect (CF).

Internet Protocol address

A numeric address that can be assigned to computers or applications.

See also *IP aliasing*.

Internode Communications facility

This module is the network transport layer for all PRIMECLUSTER internode communications. It interfaces by means of OS-dependent code to the network I/O subsystem and guarantees delivery of messages queued for transmission to the destination node in the same sequential order unless the destination node fails.

IP address

See *Internet Protocol address*.

IP aliasing

This enables several IP addresses (aliases) to be allocated to one physical network interface. With IP aliasing, the user can continue communicating with the same IP address, even though the application is now running on another node.

See also *Internet Protocol address*.

IPMI

See *Intelligent Platform Management Interface*.

JOIN (CF)

See *Cluster Join Services (CF)*.

keyword

A word that has special meaning in a programming language. For example, in an RMS configuration file, the keyword `object` identifies the kind of definition that follows.

leaf object (RMS)

A bottom object in a system graph. In the configuration file, this object definition is at the beginning of the file. A leaf object does not have children.

LEFTCLUSTER (CF)

A node state that indicates that the node cannot communicate with other nodes in the cluster. That is, the node has left the cluster. The reason for the intermediate `LEFTCLUSTER` state is to avoid the network partition problem.

See also *UP (CF)*, *DOWN (CF)*, *network partition (CF)*, *node state (CF)*.

link (RMS)

Designates a child or parent relationship between specific resources.

local area network

See *public LAN*.

local node

The node from which a command or process is initiated.

See also *remote node*, *node*.

log file

The file that contains a record of significant system events or messages. The Wizard Tools, the RMS base monitor, and RMS detectors each maintain their own log files as well.

Management Information Base

A hierarchical database of information about the local network device. The database is maintained by network management software such as an SNMP agent.

See also *Simple Network Management Protocol*.

MDS

See *Meta Data Server*.

message

A set of data transmitted from one software process to another process, device, or file.

message queue

A designated memory area which acts as a holding place for messages so they can be processed in the same order they were received.

Meta Data Server

GFS daemon that centrally manages the control information, or meta-data, of a file system.

MIB

See *Management Information Base*.

MMB

Abbreviation for Management Board, which is one of the hardware units installed in PRIMEQUEST.

mount point

The point in the directory tree where a file system is attached.

multihosting

Multiple controllers simultaneously accessing a set of disk drives.

native operating system

The part of an operating system that is always active and translates system calls into activities.

network partition (CF)

This condition exists when two or more nodes in a cluster cannot communicate over the interconnect; however, with applications still running, the nodes can continue to read and write to a shared device, compromising data integrity.

NIC switching mode

One of the LAN duplexing modes presented by GLS. The duplexed NIC is used exclusively, and LAN monitoring between the server and the switching HUB, and switchover if an error is detected are implemented.

node

A host that is a member of a cluster.

node state (CF)

Every node in a cluster maintains a local state for every other node in that cluster. The node state of every node in the cluster must be either UP, DOWN, or LEFTCLUSTER.

See also *UP (CF)*, *DOWN (CF)*, *LEFTCLUSTER (CF)*.

object (RMS)

A representation of a physical or virtual resource in the RMS configuration file or in a system graph.

See also *leaf object (RMS)*, *object definition (RMS)*, *object type (RMS)*.

object definition (RMS)

An entry in the configuration file that identifies a resource to be monitored by RMS. Attributes included in the definition specify properties of the corresponding resource.

See also *attribute (RMS)*, *object type (RMS)*.

object type (RMS)

A category of similar resources monitored as a group, such as disk drives. Each object type has specific properties, or attributes, which limit or define what monitoring or action can occur. When a resource is associated with a particular object type, attributes associated with that object type are applied to the resource.

See also *generic type (RMS)*.

online maintenance

The capability of adding, removing, replacing, or recovering devices without shutting or powering off the node.

operating system dependent (CF)

This module provides an interface between the native operating system and the abstract, OS-independent interface that all PRIMECLUSTER modules depend upon.

Oracle Real Application Clusters (RAC)

Oracle RAC allows access to all data in a database to users and applications in a clustered or MPP (massively parallel processing) platform. Formerly known as Oracle Parallel Server (OPS).

OSD (CF)

See *operating system dependent (CF)*.

parent (RMS)

An object in the RMS configuration file or system graph that has at least one child.

See also *child (RMS)*, *configuration file (RMS)*, *leaf object (RMS)*, *system graph (RMS)*.

physical IP address

The IP address that is directly assigned to the interface of a network interface card (for example, hme0).

primary node (RMS)

The default node on which a user application comes online when RMS is started. This is always the node name of the first child listed in the `userApplication` object definition.

PRIMECLUSTER services (CF)

Service modules that provide services and internal interfaces for clustered applications.

private network addresses

Private network addresses are a reserved range of IP addresses specified by the Internet Corporation for Assigned Names and Numbers (ICANN). Modern switches and routers prevent these addresses from being routed to the Internet, allowing two or more organizations to assign the same private addresses for internal use without causing conflicts or security risks.

private resource (RMS)

A resource accessible only by a single node and not accessible to other RMS nodes.

See also *resource (RMS)*, *shared resource*.

public LAN

The local area network (LAN) by which normal users access a machine.

See also *administrative LAN*.

queue

See *message queue*.

redundancy

The capability of one component to assume the resource load of another physically similar component in case the original component fails or is shut down. Common examples include RAID hardware and/or RAID software to replicate data stored on secondary storage devices, multiple network connections to provide alternate data paths, and multiple nodes that can be dynamically reprovisioned to maintain critical services in a cluster.

Reliant Monitor Services (RMS)

The package that maintains high availability of user-specified resources by providing monitoring and switchover capabilities on Linux and Solaris platforms.

remote node

A node that is accessed through a LAN or telecommunications line.

See also *local node*, *node*.

reporting message (RMS)

A message that a detector uses to report the state of a particular resource to the base monitor.

resource (RMS)

A hardware or software element (private or shared) that provides a function such as a mirrored disk, mirrored disk pieces, or a database server. A local resource is monitored only by the local node.

See also *private resource (RMS)*, *shared resource*.

resource definition (RMS)

See *object definition (RMS)*.

resource label (RMS)

The name of the resource as displayed in a system graph.

resource state (RMS)

Current state of a resource.

RMS

See *Reliant Monitor Services (RMS)*.

RMS commands (RMS)

Commands that enable RMS resources to be administered from the command line.

RMS configuration (RMS)

A configuration made up of two or more nodes connected to shared resources. Each node has its own copy of operating system and RMS software, as well as its own applications.

RMS Wizard Kit (RMS)

Each component of RMS Wizard Kit adds new menu items to the RMS Wizard Tools of the specific applications (Oracle and Networker).

See also *RMS Wizard Tools (RMS)*, *Reliant Monitor Services (RMS)*, and *wizard (RMS)*.

RMS Wizard Tools (RMS)

A software package composed of various configuration and administration tools used to create and manage applications in an RMS configuration.

See also *RMS Wizard Kit (RMS)*, *Reliant Monitor Services (RMS)*.

SAN

See *Storage Area Network*.

scalability

The ability of a computing system to efficiently handle any dynamic change in work load. Scalability is especially important for Internet-based applications where growth caused by Internet usage presents a scalable challenge.

Scalable Internet Services (SIS)

The package that dynamically balances network traffic loads across cluster nodes while maintaining normal client/server sessions for each connection.

SCON

See *single console*.

script (RMS)

A shell program executed by the base monitor in response to a state transition in a resource. The script may cause the state of a resource to change.

service node (SIS)

Service nodes provide one or more TCP services (such as FTP, Telnet, and HTTP) and receive client requests forwarded by the gateway nodes.

See also *database node (SIS)*, *gateway node (SIS)*, *Scalable Internet Services (SIS)*.

SF

See *Shutdown Facility*.

shared resource

A resource, such as a disk drive, that is accessible to more than one node.

See also *private resource (RMS)*, *resource (RMS)*.

Shutdown Facility

The PRIMECLUSTER interface that manages the shutdown of cluster nodes. The SF is automatically invoked during failover operations. It also notifies other PRIMECLUSTER products of the successful completion of node shutdown so that recovery operations can begin.

Simple Network Management Protocol

A set of protocols that facilitates the exchange of information between managed network devices. The protocols are implemented by software agents residing in the devices. Each agent can read and write data in the local Management Information Base (MIB) in response to SNMP requests from other devices on the network.

See also *Management Information Base*.

single console

The workstation that acts as the single point of administration for nodes being monitored by RMS. The single console software, SCON, is run from the single console.

SIS

See *Scalable Internet Services (SIS)*.

SNMP

See *Simple Network Management Protocol*.

state

See *resource state (RMS)*.

Storage Area Network

The high-speed network that connects multiple, external storage units and storage units with multiple computers. The connections are generally fiber channels.

subapplication (RMS)

A part of the configuration template that is designed to configure one resource type for high availability. The RMS configuration may include multiple instances of each resource type. The Generic template contains subapplications for commands, application controllers, IP addresses, virtual network interfaces, local and remote file systems, volume managers, and storage managers.

switching mode

LAN duplexing mode presented by GLS. There are switching mode types: fast switching mode, NIC switching mode, GS/SURE linkage mode (Solaris), GS linkage mode (Linux), virtual NIC mode, and multipath mode (Solaris).

switchover (RMS)

The process by which RMS switches control of a `userApplication` over from one monitored node to another.

See also *automatic switchover (RMS)*, *directed switchover (RMS)*, *failover (RMS, SIS)*, *symmetrical switchover (RMS)*.

symmetrical switchover (RMS)

This means that every RMS node is able to take on resources from any other RMS node.

See also *automatic switchover (RMS)*, *directed switchover (RMS)*, *failover (RMS, SIS)*, *switchover (RMS)*.

synchronized power control

When the power of one node is turned in the cluster system, this function turns on all other powered-off nodes and disk array unit that are connected to nodes through RCI cables.

system disk (GDS)

Disk on which the active operating system is installed. System disk refers to the entire disk that contains the slices that are currently operating as one of the following file systems (or the swap area):

For Solaris: /, /usr, /var, or swap area

For Linux: /, /usr, /var, /boot, /boot/efi, or swap area

system graph (RMS)

A visual representation (a map) of monitored resources used to develop or interpret the RMS configuration file.

See also *configuration file (RMS)*.

template

See *application template (RMS)*.

type

See *object type (RMS)*.

UP (CF)

A node state that indicates that the node can communicate with other nodes in the cluster.

See also *DOWN (CF)*, *LEFTCLUSTER (CF)*, *node state (CF)*.

virtual disk

A pseudo-device that allows a portion or a combination of physical disks to be treated as a single logical disk. The virtual disk driver is inserted between the highest level of the OS logical input/output (I/O) system and the physical device driver(s), allowing all logical I/O requests to be mapped to the appropriate area on the physical disk(s).

Glossary

See also *subapplication (RMS)*.

Web-Based Admin View

A Java-based, OS-independent interface to PRIMECLUSTER management components.

See also *Cluster Admin*.

wizard (RMS)

An interactive software tool that creates a specific type of application using pretested object definitions.

Wizard Kit (RMS)

See *RMS Wizard Kit (RMS)*.

Wizard Tools (RMS)

See *RMS Wizard Tools (RMS)*.

Abbreviations

AC	Access Client
API	application program interface
AS	Adaptive Services
bm	base monitor
CCBR	Cluster Configuration Backup/Restore
CDL	Configuration Definition Language
CF	Cluster Foundation or Cluster Framework
CIM	Cluster Integrity Monitor
CIP	Cluster Interconnect Protocol
CLI	command line interface
CLM	Cluster Manager
CRM	Cluster Resource Management
DHCP	Dynamic Host Control Protocol
DLPI	Data Link Provider Interface
ELM	Enhanced Lock Manager

Abbreviations

ENS	Event Notification Services
GDS	Global Disk Services
GFS	Global File Services
GLS	Global Link Services
GUI	graphical user interface
HA	high availability
ICF	Internode Communication Facility
I/O	input/output
IPMI	Intelligent Platform Management Interface
JOIN	cluster join services module
LAN	local area network
MDS	Meta Data Server
MIB	Management Information Base
MIPC	Mesh Interprocessor Communication
MMB	Management Board (on PRIMEQUEST systems)
NIC	network interface card

NSM	Node State Monitor
OSD	operating system dependent
PAS	Parallel Application Services
RCCU	Remote Console Control Unit
RCI	Remote Cabinet Interface
RMS	Reliant Monitor Services
SA	Shutdown Agent
SAN	Storage Area Network
SCON	single console software
SD	Shutdown Daemon
SF	Shutdown Facility
SIS	Scalable Internet Services
SNMP	Simple Network Management Protocol
VIP	Virtual Interface Provider
XSCF	eXtended System Control Facility

Abbreviations

Figures

Figure 1:	Overview of PRIMECLUSTER products	10
Figure 2:	Interface between RMS and the operating system	14
Figure 3:	Parent application with two dependencies	17
Figure 4:	RMS representation of controlled application	18
Figure 5:	Result of follow-mode switchover	19
Figure 6:	Scalable mode child (controlled) application switchover	20
Figure 7:	Scalable mode parent (controlling) application switchover	20
Figure 8:	Controller structure with multiple children	21
Figure 9:	Relationship between RMS and RMS Wizards	23
Figure 10:	Object hierarchy for initializing examples	38
Figure 11:	System graph for initializing examples—RMS Wizard Tools	39
Figure 12:	hvdsp output for initializing examples—RMS Wizard Tools	40
Figure 13:	Example of maintenance mode restrictions	63
Figure 14:	Scalable controller attributes	74
Figure 15:	Maintenance mode controller example	83
Figure 16:	Switching scalable parent with child in maintenance mode	84
Figure 17:	Switching scalable child with parent in maintenance mode	85
Figure 18:	Attempting to take a parent offline with the child in maintenance mode	86
Figure 19:	Attempting to take a child offline with the parent in maintenance mode	87
Figure 20:	Main configuration menu when RMS is not active	94
Figure 21:	Main configuration menu when RMS is running	99
Figure 22:	Application type selection	100
Figure 23:	Menu leading to basic settings	100

Figures

Figure 24:	Menu to configure basic settings	101
Figure 25:	Menu to configure non-basic settings	101
Figure 26:	Main configuration menu	102
Figure 27:	Activating a configuration	103
Figure 28:	Quitting the Main configuration menu	104
Figure 29:	Main configuration menu	108
Figure 30:	Application selection menu	109
Figure 31:	Option menu of the application setting	109
Figure 32:	Application selection menu	109
Figure 33:	Application setting menu	110
Figure 34:	Selection menu of the PreCheckScript	110
Figure 35:	Entering PreCheckScript	111
Figure 36:	Application setting menu	111
Figure 37:	Application selection menu	112
Figure 38:	Main configuration menu	112
Figure 39:	Generation and distribution of the RMS configuration file	113
Figure 40:	Quitting main configuration menu	113
Figure 41:	Main configuration menu	114
Figure 42:	Application selection menu	114
Figure 43:	Option menu of application setting	115
Figure 44:	Application selection menu	115
Figure 45:	Settings menu of controller	116
Figure 46:	Selection menu of controlled target application	116
Figure 47:	Attribution setting menu of the controller	116
Figure 48:	TIMEOUT input	117
Figure 49:	Attribution setting menu of the controller	117

Figure 50:	Menu settings of the controller	117
Figure 51:	Application selection menu	118
Figure 52:	Main configuration menu	118
Figure 53:	Creation and distribution of designated file of RMS configuration 119	
Figure 54:	Quitting main configuration menu	119
Figure 55:	Web-Based Admin View login	123
Figure 56:	Invoking the Cluster Services GUI	124
Figure 57:	Invoking Cluster Admin	124
Figure 58:	Cluster Admin initial connection menu	125
Figure 59:	Main Cluster Admin window—Initial view	126
Figure 60:	Main Cluster Admin window—message view	127
Figure 61:	Main Cluster Admin window—RMS view	128
Figure 62:	RMS tree with a controller object	130
Figure 63:	Pop-up context menu for a node	131
Figure 64:	Pop-up context menu for an online application	132
Figure 65:	Pop-up context menu for an offline application	132
Figure 66:	Pop-up confirmation dialog	133
Figure 67:	Displaying global environment variables	133
Figure 68:	Global environment variable view	134
Figure 69:	Displaying local environment variables	134
Figure 70:	Local environment variables view	135
Figure 71:	Displaying application and object states	136
Figure 72:	Configuration information or object attributes	137
Figure 73:	Opening the clusterwide table	139
Figure 74:	Clusterwide table	139

Figures

Figure 75:	Clusterwide table with state names	140
Figure 76:	Faulted applications in the clusterwide table	140
Figure 77:	Offline applications in the clusterwide table	141
Figure 78:	Faulted and offline applications in the clusterwide table	141
Figure 79:	Split-brain conditions in the clusterwide table	142
Figure 80:	Using context menus in the clusterwide table	142
Figure 81:	Viewing the RMS full graph on a node	144
Figure 82:	RMS full graph on a node—tab view	145
Figure 83:	Detail of tab view showing detach button	145
Figure 84:	RMS full graph on a node—separate window view	146
Figure 85:	Detail of separate window view showing attach button	146
Figure 86:	RMS full graph—object tooltip	147
Figure 87:	RMS full graph—object details	148
Figure 88:	Viewing an RMS application graph	148
Figure 89:	Typical RMS application graph	149
Figure 90:	Viewing an RMS subapplication graph	150
Figure 91:	Typical RMS subapplication graph	150
Figure 92:	Standard view of configuration with controlled application	151
Figure 93:	Viewing an RMS composite subapplication graph	152
Figure 94:	Typical composite subapplication graph	152
Figure 95:	Using a pop-up context menu from the RMS graph	153
Figure 96:	Displaying an RMS graph with affiliation names	154
Figure 97:	RMS graph with resource names	155
Figure 98:	RMS graph with resource names	156
Figure 99:	RMS graph with resource and affiliation names	157
Figure 100:	Sorting object names in the graph	157

Figure 101: Cluster state before RMS is shut down	158
Figure 102: Cluster state after RMS restart with different configuration	159
Figure 103: RMS main view, graphs, and clusterwide table after shutdown on one node	160
Figure 104: Application graph on shutdown node	160
Figure 105: Graph window for deleted object	161
Figure 106: Viewing the RMS switchlog file using a context menu . . .	162
Figure 107: Viewing the RMS switchlog file using the Tools menu . . .	162
Figure 108: Viewing an application log using a context menu	163
Figure 109: RMS switchlog in tab view	164
Figure 110: Detail of tab view showing detach button	164
Figure 111: RMS switchlog in detached view	165
Figure 112: Detail of detached window view showing attach button . .	165
Figure 113: Search based on date and time range	167
Figure 114: Search based on resource name	168
Figure 115: Search based on severity level	169
Figure 116: Search based on keyword	170
Figure 117: Using the pop-up Find dialog in log viewer	171
Figure 118: Starting RMS from the main menu	174
Figure 119: RMS Start Menu for all the nodes	174
Figure 120: RMS Start Menu for individual nodes	175
Figure 121: Starting RMS on individual nodes	176
Figure 122: Controlling automatic RMS startup—step 1	179
Figure 123: Controlling automatic RMS startup—step 2	180
Figure 124: Using the Tools menu to stop RMS	181
Figure 125: Stopping RMS on all available nodes	182

Figures

Figure 126: Stopping RMS on one node from the list	183
Figure 127: Stopping RMS while keeping applications—confirmation	184
Figure 128: Forced shutdown of RMS—confirmation	184
Figure 129: Using the context menu to stop RMS on one node	184
Figure 130: Stopping RMS on one node	185
Figure 131: Controlling automatic application startup—step 1	189
Figure 132: Controlling automatic application startup—step 2	189
Figure 133: Switching an application	191
Figure 134: Switching a busy application	192
Figure 135: Starting an application	193
Figure 136: Shutting down an application	194
Figure 137: Resetting an application	195
Figure 138: Choosing the reset timeout for the application	196
Figure 139: Clearing an application fault	197
Figure 140: Clearing an application fault—confirmation dialog	197
Figure 141: Switching a resource	200
Figure 142: Confirming a resource switchover	200
Figure 143: Starting a resource	201
Figure 144: Confirming a resource start	202
Figure 145: Stopping a resource	203
Figure 146: Confirming a resource stop	203
Figure 147: Checking the failed resource	205
Figure 148: Displaying Fault Occurred in the StateDetails attribute . .	206
Figure 149: Clearing the Fault trace	207
Figure 150: Checking whether the Faulted trace is cleared	207
Figure 151: Starting maintenance mode for all applications	208

Figure 152: Starting maintenance mode for a single application	209
Figure 153: Maintenance mode confirmation for all applications	209
Figure 154: Maintenance mode confirmation for one application	209
Figure 155: Typical cluster in maintenance mode	210
Figure 156: Normal operation of independent application	211
Figure 157: Normal maintenance mode exit for all applications	212
Figure 158: Normal maintenance mode exit for a single application . .	212
Figure 159: Forced maintenance mode confirmation for all applications	213
Figure 160: Wizard Tools Detector Details menu	270

Index

Symbols

- #RMS# entries
 - /etc/fstab.pcl 221
- >> input prompt 93

A

- activating
 - and hvipalias file 220
 - applications 198
 - configuration 96, 102
 - configurations 173
- administrative privileges
 - Web-Based Admin View login 123
- administrator prerequisites 1
- Affiliation attribute 241
- alerts
 - log viewer 169
- Alternatelp attribute 240
- andOp objects
 - defined* 231
- applets
 - Java, trusted 125
- application logs
 - searching text 171
 - viewing 163
- applications
 - activating 198
 - as objects 13
 - autostartup, overriding 188
 - clearing faults 196
 - deactivating 198
 - dependencies 129
 - displaying states 135
 - failover 16
 - going offline 48
 - log files 267
 - maintenance mode, clusterwide 210
 - objects, overview 106
 - offline 194
 - online 193
 - overriding AutoStartUp 252

Index

- resetting, GUI 195
 - resetting, hvreset command 196
 - starting, GUI 193
 - starting, hvswitch command 193
 - stopping, GUI 194
 - stopping, hvshut command 185
 - switching 60
 - switching to Standby state 194
 - switching to SysNode 29, 285
 - switching, GUI 190
 - switching, hvswitch command 192
 - viewing attributes 137
 - viewing graph 148
 - viewing log files 162
 - viewing logs 163
- attach button
- graph window 146, 165
- attributes
- defined* 30
 - Affiliation 241
 - Alternatelp 240
 - AutoRecover 233
 - AutoRecoverCleanup 241
 - AutoStartUp 233
 - AutoSwitchOver 233
 - Class 241
 - ClusterExclusive 234
 - Comment 241
 - ControlledSwitch 241
 - DetectorStartScript 241
 - ENV object 231
 - FaultScript 234
 - for andOp objects 231
 - for gController objects 231
 - for gResource objects 231
 - for orOp objects 231
 - for SysNode objects 232
 - for userApplication objects 232
 - Halt 234
 - HostName 242
 - I_List 242
 - LastDetectorReport 242
 - LieOffline 235
 - MaxControllers 242

- MonitorOnly 235
- NoDisplay 242
- NullDetector 242
- OfflineDoneScript 235
- OfflineScript 235
- OnlinePriority 236
- OnlineScript 236
- PartialCluster 237
- PersistentFault 237
- PostOfflineScript 237
- PostOnlineScript 237
- PreCheckScript 238
- PreOfflineScript 238
- PreOnlineScript 239
- PreserveState 239
- PriorityList 243
- Resource 243
- rName 243
- ScriptTimeout 239
- ShutdownPriority 239
- SplitRequest 243
- StandbyCapable 240
- StateDetails 244
- viewing, for objects 137
- WarningScript 240
- authentication
 - trusted 217
- automatic RMS startup 179, 181
- AutoRecover attribute
 - defined* 233
 - fault processing 55
- AutoRecoverCleanup attribute 241
- AutoStartUp attribute
 - defined* 233
 - overriding 188, 252
- AutoSwitchOver attribute
 - defined* 233
 - fault processing 52, 53
- B**
- base monitor
 - defined* 25
 - cluster monitoring 249
 - communication port 248

Index

- debug messages 261
- detectors 105
- heartbeat 279
- high availability 12
- locking process in memory 253
- log file 263
- log levels 265
- messages 263
- object states 227
- setting log levels 264
- stack tracing 265
- switchlog 266
- basic settings
 - Wizard Tools 100
- bin
 - RMS directory 33
- bmlog description 263
- browser
 - starting Cluster Admin 122
- build
 - RMS directory 33
- buttons
 - attach, graph window 146, 165
 - detach, graph tab view 145, 164
 - Filter, log viewer 166

C

- case-insensitive
 - searching logs 170
- case-sensitive
 - searching logs 171
- Caution
 - defined* 6
 - Do not explicitly set RMS environment variables in the user environment 32, 245
 - If the HV_AUTOSTARTUP_IGNORE environment variable is used, ensure that it is correctly defined on all cluster nodes and that it is always kept up-to-date 247
 - Leaving applications running after stopping RMS can lead to data corruption 182
 - Leaving applications running after stopping RMS or using forced shutdown can cause data corruption 183
 - Use care when clearing a SysNode Wait state manually 187

-
- Use hvshut -A, -f, and -L options carefully as they could result in inconsistencies or data corruption 186
 - Use the forced switch mode only if an application cannot be switched normally 191
 - Use the hvswitch -f option carefully as it could result in inconsistencies or data corruption 192
- CF
- defined* 10
 - Cluster Admin view 126
 - configuration 129
 - LEFTCLUSTER 59
 - node names 121, 219
 - required for ELM 249
 - tabbed view, Cluster Admin 126
 - trusted login 218
- cftool command 219
- changing
- configurations, and clusterwide table 158
 - detail level in graphs 154
 - environment variables 31, 245
 - HV_AUTOSTARTUP 190
 - HV_RCSTART 180
- CIP
- RMS node names 216
- Class attribute 241
- clearing
- faults, GUI 196
 - faults, hvutil command 197
 - hung nodes 285
 - resource faults 285
 - SysNode Wait state, GUI 186
- CLI *see* RMS commands
- CLI *see* RMS commands
- closely coupled applications 19
- cluster 1
- high availability 10
 - monitoring by RMS 249
 - services 9
 - switching userApplication 27
- Cluster Admin 25
- administrative privileges 123
 - application graph 148
 - clusterwide table 139
 - context menus 131

- graph 151
- GUI 121
- initial view 125
- logging in 123
- management server 122
- object attributes 137
- operator privileges 123
- overview 10
- pop-up menus 131
- RMS graphs 143
- RMS tree 129
- root privileges 123
- searching log text 171
- starting 122
- switchlog 162
- SysNode selection 131
- userApplication selection 131
- using 122
- viewing log files 162
- cluster file system 9
- Cluster Foundation 66
- Cluster Foundation *see* CF
- cluster nodes *see* nodes
- cluster volume management 9
- ClusterExclusive attribute
 - defined* 234
- clusterwide states
 - and maintenance mode 210
- clusterwide table
 - defined* 139
 - context menus 142
 - displaying state names 140
 - opening 139
 - special display order 141
- Cmdline, resource wizard 91
- command menus *see* context menus
- commands
 - hvdump 280
- commands *see* RMS commands
- commands *see* RMS commands
- Comment attribute 241
- communication port
 - base monitor 248
- composite subapplication graph 151

-
- CONFIG.rms
 - default RMS startup file 177, 180, 253
 - configuration tree
 - RMS, in Cluster Admin 129
 - configurations
 - defined* 11
 - activating 173
 - displaying 28, 283
 - displaying information 126, 128
 - general procedure 91
 - graph 143
 - viewing information 137
 - configuring
 - applications 89, 91
 - file systems 91
 - IP addresses 91
 - resources 91
 - confirming
 - object actions 133
 - context menus
 - closing 131, 143, 153
 - Cluster Admin 131
 - clusterwide table 142
 - environment variables 133
 - equivalence 163
 - from Cluster Admin 126
 - graph 153
 - RMS graph 153
 - RMS tree 131
 - stopping RMS 184
 - switching applications 191
 - viewing application graph 148
 - viewing clusterwide table 139
 - viewing RMS graph 143
 - viewing subapplication graph 149
 - viewing switchlog 162
 - controlled applications
 - description 16
 - failover 18
 - ControlledSwitch attribute 241
 - Controller objects
 - resource wizard 91
 - Scalable mode 19
 - controller objects

Index

- see also* controllers
 - defined* 231
 - dependencies 129
 - description 16
 - follow mode 18
 - graph 152
 - overview 106
- controlling
 - RMS operation 173
- critical errors
 - log viewer 169
- cron jobs
 - hvlogcontrol 275
- D**
- Deact state 227
- deactivating
 - applications 198
- debug level, wizards 271
- debug messages 259
 - base monitor 261
 - log directory 261
 - log viewer 169
- debug reporting, wizards 270
- defining
 - shutdown timeout 250
- deleting
 - Cluster Admin messages 127
 - log files 284
 - log files by size 252
- DEMO turnkey wizard 93
- dependencies
 - applications 129
- dependent resources
 - for application 13
- detach button
 - graph tab view 145, 164
- details
 - changing level of, in graphs 154
- detector
 - trace, set automatically 279
- detectors
 - defined* 12
 - fault situations 47, 51

-
- illegal 37
 - log files 267, 268
 - names 268
 - provided by RMS 26
 - RMS Wizard Kit 24
 - RMS Wizard Tools 24
 - setting log levels with hvutil -L 269
 - setting log levels with Wizard Tools 270
 - starting 176
 - switchlog file 267
 - DetectorStartScript attribute 241
 - device numbers
 - shared NFS file systems 225
 - dialogs
 - confirmation 133
 - find, log viewer 171
 - directed switch requests 60
 - directories
 - RMS 33
 - RMS, specifying root 250
 - disk classes
 - as RMS resources 22
 - displaying
 - see also* viewing 128
 - application states 135, 136
 - current RMS configuration 28, 283
 - environment variables 133, 135
 - HV_AUTOSTARTUP 190
 - HV_RCSTART 180
 - object states 135, 136
 - state names, clusterwide table 140
 - DNS
 - and cluster host names 216
 - and /etc/fstab.pcl entries 221
 - documentation
 - related 2
 - double faults 53
 - defined* 234
 - and Halt attribute 53, 234
 - E**
 - echo service 226
 - ELM
 - defined* 66
-

- enabling and disabling 249
- emergency
 - log viewer 169
- Enhanced Lock Manager *see* ELM
- ENV object 31, 231
- environment variables
 - categories* 31
 - displaying 133, 135
 - HV_APPLICATION 79, 256
 - HV_APPLICATION_STATE_CHANGE_INFO, scalable controller script 78
 - HV_AUTORECOVER 256
 - HV_AUTORECOVER script AutoRecover attempt flag 79
 - HV_AUTOSTART_WAIT 247
 - HV_AUTOSTARTUP 252
 - HV_AUTOSTARTUP_IGNORE 246
 - HV_CHECKSUM_INTERVAL 247
 - HV_COM_PORT 248
 - HV_CONNECT_TIMEOUT 252
 - HV_FORCED_REQUEST 256
 - HV_FORCED_REQUEST script forced request flag 79
 - HV_HOST_STATE_CHANGE_INFO, scalable controller script 78
 - HV_LAST_DET_REPORT 256
 - HV_LAST_DET_REPORT,
 - script last detector report 79
 - HV_LOG_ACTION 252
 - HV_LOG_ACTION_THRESHOLD 248
 - HV_LOG_WARN_THRESHOLD 248
 - HV_MAXPROC 253
 - HV_MLOCKALL 253
 - HV_NODENAME 256
 - HV_NODENAME,
 - script current object name 79
 - HV_OFFLINE_REASON 256
 - HV_OFFLINE_REASON,
 - script reason for offline processing 79
 - HV_RCSTART, changing 180
 - HV_RCSTART, *defined* 253
 - HV_REALTIME_PRIORITY 253
 - HV_REQUESTING_CONTROLLER script forced request flag 256
 - HV_SCALABLE_CONTROLLER, scalable controller script 257
 - HV_SCALABLE_INFO, scalable controller script 257
 - HV_SCRIPT_TYPE 257

-
- HV_SCRIPT_TYPE,
 - script type 79
 - HV_SCRIPTS_DEBUG 254
 - HV_SYSLOG_USE 254
 - HV_USE_ELM 249
 - NODE_SCRIPTS_TIME_OUT 257
 - NODE_SCRIPTS_TIME_OUT, script timeout value for current object 79
 - RELIANT_HOSTNAME 254
 - RELIANT_INITSCRIPT 255
 - RELIANT_LOG_LIFE 250
 - RELIANT_LOG_PATH 250, 261
 - RELIANT_PATH 250
 - RELIANT_SHUT_MIN_WAIT 250
 - RELIANT_STARTUP_PATH 255
 - SCRIPTS_TIME_OUT 255
 - ENVL object 31, 231
 - error levels
 - log viewer 169
 - error messages 259
 - base monitor 261
 - switchlog, script timeout 60
 - errors
 - at initialization 48
 - during offline processing 51
 - in offline state 54
 - reaction to 52
 - etc
 - RMS directory 33
 - /etc/dfs/dfstab.pcl 223
 - /etc/fstab.pcl
 - format* 221
 - conflicts in /etc/fstab 221
 - /etc/hosts
 - and /etc/fstab.pcl entries 221
 - and network interface names 219
 - entries for RMS 216
 - /etc/nsswitch.conf 226
 - /etc/vfstab.pcl 223
 - exit codes
 - RMS CLI 122
 - RMS scripts 12
 - expert operations
 - disabling ELM 249

Index

F

failover

- defined* 66
- applications 16
- controlled applications 18
- directed by RMS 16
- NFS file systems 225
- NFS Lock Failover 224
- nodes 16

fatal errors

- log viewer 169

fault script

- RMS reaction to fault 52

Faulted state

- defined* 227
- clearing 58, 196

faults

- clearing 196
- display in clusterwide table 141
- messages, custom handling 220
- SysNode 59

FaultScript attribute 234

file handle

- NFS file systems 225

file systems

- as application resources 89
- as RMS resources 22
- filling up 271
- Fsystem 91
- NFS failover 225
- resource type 24
- site preparation 215
- warning threshold 248

Filter button

- log viewer 166

filters

- log viewer 166

follow controllers

- described 18

forcing

- offline operations, and status code 122
- RMS shutdown, GUI 183
- RMS shutdown, hvshut command 185

fstab file *see* /etc/fstab.pcl

Fsystem, resource wizard 91

G

Gds, resource wizard 91

Global Disk Services 91

global environment variables 31

Gls, resource wizard 91

graphs

defined 11

appearance after shut down 159

application 148

application, resetting 195

changing detail level 154

composite subapplication 151

context menus 153

detaching view 146, 165

internal, created by generate phase 96

maintenance mode scope 211

reinitializing 284

subapplications 149

tabbed view 145, 164

tabbed view, Cluster Admin 126

types 143

viewing application 148

viewing object details 147

gResource objects

defined 231

overview 106

representing physical resources 13

GUI, Cluster Admin

configuration tree 126

input and message area 126

menu bar 126

messages 127

starting RMS 128

H

Halt attribute

defined 234

and double faults 53

heartbeat, base monitor 279

heartbeats

and SysNode faults 59

cluster monitoring 65

Index

- mode 249
 - recovery timeout 176
 - UDP, and ELM 249
- high availability 1
 - provided by RMS 9
 - specifying applications 95
- HostName attribute
 - defined* 242
 - andOp objects 231
- hosts
 - preparing for RMS 215
- hosts file *see* /etc/hosts
- HV_APPLICATION 79, 256
- HV_APPLICATION_STATE_CHANGE_INFO 78
- HV_AUTORECOVER 79, 256
- HV_AUTOSTART_WAIT 247
- HV_AUTOSTARTUP
 - defined* 252
 - AutoStartUp attribute 233
 - changing at runtime 188
- HV_AUTOSTARTUP_IGNORE 246
- HV_CHECKSUM_INTERVAL 247
- HV_COM_PORT 248
- HV_CONNECT_TIMEOUT 252
- HV_FORCED_REQUEST 79, 256
- HV_HOST_STATE_CHANGE_INFO 78
- HV_LAST_DET_REPORT 79, 256
- HV_LOG_ACTION 252, 275
- HV_LOG_ACTION_THRESHOLD 248, 275
- HV_LOG_WARN_THRESHOLD 248, 275
- HV_MAX_HVDISP_FILE_SIZE 252
- HV_MAXPROC 253
- HV_MLOCKALL 253
- HV_NODENAME 79, 256
- HV_OFFLINE_REASON 79, 256
- HV_RCSTART
 - defined* 253
 - changing 180
- HV_REALTIME_PRIORITY 253
- HV_REQUESTING_CONTROLLER 256
- HV_SCALABLE_CONTROLLER 257
- HV_SCALABLE_INFO 257
- HV_SCRIPT_TYPE 79, 257
- HV_SCRIPTS_DEBUG 254

HV_SYSLOG_USE 254
HV_USE_ELM 249
hvassert command 28
 defined 283
hvcm command 28
 defined 283
 -h option 249
 starting RMS 176
hvconfig command 28
 defined 283
hvconsoles file 220
hvdisp command
 -a option 136
 displaying environment variables 135, 246
 displaying object states 136
 ENV and ENVL 135
 file size 253
hvdump command 28, 280
 defined 284
hvexec command 105
hvipalias file
 format 219
 activation 220
hvmlogclean command 28, 275
 defined 284
hvmlogcontrol command 275
 log file cleanup 248
hvreset command
 defined 284
 resetting applications 196
 -t option 196
hvsetenv command 29
 defined 284
 HV_AUTOSTARTUP 190, 252
 HV_RCSTART 180, 253
hvshut command 29
 defined 285
 defining timeout 250
 stopping RMS 185
hvswitch command 29
 defined 285
 starting application 193
 switching application 192
hvutil

Index

- l base monitor log level 264
- L, setting detector log levels 269
- hvutil command
 - defined* 285
 - a option 198
 - activating applications 198
 - c option 58, 197
 - clearing faults 197
 - clearing Wait state 187
 - d option 198
 - deactivating applications 198
 - defined* 29
 - f option 194
 - m and -M options 213
 - maintenance mode 213
 - o option 187
 - s option 194
 - Standby state 194
 - stopping applications 194
- hvw command
 - defined* 92
 - operation mode 97
 - provided by RMS Wizard Tools 24
- I**
- I_List attribute 242
- icons, Cluster Admin
 - application status 210
 - legend 135
 - maintenance mode 210
 - object status 135, 141
- icons, clusterwide table
 - object status 140, 141, 142
- icons, RMS graph
 - object type 146
- include
 - RMS directory 33
- Inconsistent state
 - defined* 227
- informational messages
 - log viewer 169
- initialization
 - error during 48
 - initial object state 37

- objects 37
 - script, specifying 255
 - Unknown state 37
- InitScript 26
- intended state
 - maintenance mode 210, 228
- IP addresses
 - as RMS resources 22
 - not allowed in /etc/fstab.pcl 221
 - resource wizard 91
- IP alias, logical interface name 219
- Ip Address subapplication
 - and /etc/hosts 217
 - hvipalias file 219
- Ipaddress, resource wizard 91
- IPV6 addresses
 - RMS support 216
- J**
- Java
 - client required for Windows 122
 - trusted applets 125
- K**
- kernel
 - ELM locks 66
- killing a node 13, 16
- L**
- LastDetectorReport attribute 242
- left pane, Cluster Admin
 - context menu 139
 - object states 135
 - RMS tree 126, 128
 - tabs 126
- LEFTCLUSTER
 - CF event, SysNode fault 59
- legend
 - Cluster Admin icons 135
- lib
 - RMS directory 33
- LieOffline attribute 235
- local controllers
 - maintenance mode 211

Index

- local environment variables 31
- locking
 - base monitor in memory 253
- locks, managed by ELM 66
- log file cleanup 275
- log files
 - applications 267
 - base monitor 263
 - custom fault messages 220
 - detectors 267, 268
 - location 34
 - searching 171
 - specify directory 250
 - switchlog 60, 225, 262
 - system 225
 - time of preservation 250
 - viewing 162, 163
 - Wizard Tools detectors 268
- log levels
 - controlling 264
 - detectors, setting in Wizard Tools 270
 - detectors, setting with hvutil -L 269
 - detectors, setting with Wizard Tools 270
- log messages, wizards 266
- log viewer
 - described 162
 - error levels 169
 - filters 166
- logical interface name, IP alias 219
- login
 - Cluster Admin 123
 - trusted, required by Wizard Tools 217

M

- main menu
 - Wizard Tools 94
- maintenance mode
 - defined* 208
 - CLI operations 213
 - entering 208
 - exiting 211
 - intended state 210
 - local controllers 211
 - Maintenance state 228

- operating notes 211
- setting 285
- starting 208
- status code when entering 122
- stopping 211
- man page, for RMS commands 27
- management server, Web-Based Admin View 122
- market-specific
 - applications 10
- MaxControllers attribute 242
- messages
 - base monitor 263
 - between RMS objects 36
 - bmlog 263
 - debug 261
 - error 261
 - fault, custom handling 220
 - generic detector log 262
 - troubleshooting RMS 276
 - wizards 266
- MonitorOnly attribute
 - defined* 235
- N**
- naming conventions
 - RMS 216
- netmask
 - in hvipalias file 219
- network interfaces
 - in /etc/hosts 217
 - in hvipalias file 219
 - site preparation 215
- NFS file systems
 - failover 225
 - Lock Failover 224
- node
 - Cluster Admin connection 124
 - Web-Based Admin View connection 122
- node names
 - CF 121
 - in configuration files 216
 - RMS 216
- NODE_SCRIPTS_TIME_OUT 79, 257
- nodes

Index

- detector timeout for remote 252
- failover 16
- ignore at startup 246
- killing 13, 16
- objects, overview 106
- primary 140
- viewing attributes 137
- wait to report online 247

NoDisplay attribute 242

non-basic settings, Wizard Tools 100

non-fatal errors

- log viewer 169

notices

- log viewer 169

NullDetector attribute 242

O

object states

- RMS graph 146
- timeouts, tracking by base monitor 11

object types

- andOp 231
- ENV 231
- ENVL 231
- gController 231
- gResource 231
- orOp 231
- SysNode 232
- userApplication 232

objects

- activating applications 198
- and scripts 12
- attributes 30
- clearing a fault 196
- confirming actions 133
- context menus 131, 142, 153
- controller 16, 17, 129, 152
- controller, overview 106
- dependencies 146
- displaying states 135
- environment variables 31
- graph 153
- graph customization 154
- gResource 13

- gResource, overview 106
- relationships 146
- resetting applications 195
- resource types 30
- RMS full graph 146
- RMS tree 131, 142
- selecting 131
- starting applications 193
- switching applications 191
- SysNode 13, 131
- SysNode, overview 106
- taking applications offline 194
- userApplication 13, 131, 186
- userApplication, overview 106
- viewing attributes 137
- viewing details in graphs 147
- offline processing
 - defined* 13, 48
 - fault situations 51
 - requests 48
 - scripts 24
- Offline state
 - defined* 227
 - general description 11
- OfflineDoneScript attribute 235
- OfflineFault state 227
- OfflineScript attribute 235
- online processing 13
- online scripts
 - provided by Wizard Tools 24
- Online state
 - defined* 227
 - general description 11
- OnlinePriority attribute 236
- OnlineScript attribute 236
- operating system
 - administrator prerequisites 1
- operator
 - intervention 60
- operator privileges
 - Web-Based Admin View login 123
- /opt/SMAW/SMAWRrms/ 33
- /opt/SMAW/SMAWRrms/etc/ 177
- /opt/SMAW/SMAWRrms/etc/CONFIG.rms 177

Index

overriding

 automatic application startup 188

P

parallel application support 9

PartialCluster attribute

defined 237

PersistentFault attribute 237

physical disks

 state at initialization 47

physical interfaces

 IP aliases 219

pop-up menus *see* context menus

PostOfflineScript attribute 237

PostOnlineScript attribute 237

PreCheckScript attribute 238

PreOffline processing 49

PreOfflineScript attribute 238

PreOnlineScript attribute 239

PreserveState attribute

defined 239

 effect on fault processing 52, 53, 54

primary management server, Web-Based Admin View 122

PRIMECLUSTER 9

PRIMECLUSTER product family

 administrator prerequisites 1

 overview 9

priority switch

 request 60

PriorityList attribute

defined 243

privileges

 Web-Based Admin View login 123

proactive scripts 12

processes

 as RMS resources 22

R

reactive scripts 12

recovery timeout

 UDP heartbeats 65, 249

related documentation 2

Reliant Monitor Services

see RMS

-
- RELIANT_HOSTNAME 254
 - RELIANT_INITSCRIPT 255
 - RELIANT_LOG_LIFE 250
 - RELIANT_LOG_PATH 34, 250, 261
 - RELIANT_PATH 33, 250
 - RELIANT_SHUT_MIN_WAIT 250
 - RELIANT_STARTUP_PATH 177, 255
 - requests
 - in RMS messages 36
 - offline processing 48
 - request-triggered scripts
 - InitScript 26
 - OfflineScript 26
 - OnlineScript 27
 - PreCheckScript 26
 - PreOnlineScript 27
 - resetting
 - applications, GUI 195
 - applications, hvreset command 196
 - Resource attribute
 - /defined* 243
 - controller objects 231
 - resource wizards
 - Cmdline 91
 - Controller 91
 - Fsystem 91
 - Gds 91
 - Gls 91
 - Ipaddress 91
 - Vxvm 91
 - resources
 - clearing faulted 285
 - configuring 91
 - defining 22
 - executing scripts 98
 - file system entries 221, 223
 - monitoring 105
 - network interfaces 219
 - non-basic settings 101
 - object types 30
 - objects, overview 106
 - scripts 26
 - shared remote entries 223
 - states 11

Index

- /.rhosts 217
- right pane
 - Cluster Admin 126
 - log messages 164
- right-click, mouse
 - Cluster Admin view 131
 - clusterwide table 142
 - graph 153
- rKind attribute
 - gResource objects 231, 243
- RMS
 - base monitor 11
 - clusterwide table 139
 - components 25
 - configuration tree, Cluster Admin 129
 - controlling operation 173
 - default directory 33
 - directory structure 33
 - executing commands at startup 26
 - full graph 143
 - graphs 96, 143
 - high availability 11
 - naming conventions 216
 - node names 121, 216
 - overview 10
 - severity levels 169
 - shutdown, and graph appearance 159
 - starting 173
 - starting, and HV_AUTOSTARTUP 188
 - stopping 181
 - tree, Cluster Admin 129
- RMS commands 27
 - see also* individual commands
 - basic list* 283
 - administration procedures 121
 - hvassert 28, 283
 - hvcn 28, 176, 283
 - hvconfig 28, 283
 - hvdsp 28, 135, 136, 284
 - hvdump 28, 284
 - hvexec 105
 - hvlogclean 28, 275, 284
 - hvlogcontrol 275
 - hvreset 196, 284

- hvsetenv 29, 180, 190, 284
- hvshut 29, 185, 186, 285
- hvswhitch 29, 192, 193, 285
- hvutil 29, 58, 187, 194, 197, 198, 213, 285
- options 121
- status codes 122
- RMS Wizard Kit
 - defined* 10
 - components 23, 24
 - detectors 24
 - hw command 24
 - scripts 24
- RMS Wizard Tools
 - defined* 10
 - and RMS Wizard Kit 22
 - configuring 22
 - detectors 24
 - hw command 24
 - resource types 24
 - scripts 24
- rName attribute
 - gResource objects 231, 243
- rolling upgrades
 - disabling ELM 67
- root privileges
 - Web-Based Admin View 123
- /root/.rhosts 217

S

- scalability 9, 19
- scalable controllers 19
- Scalable Internet Services *see* SIS
- script top-of-tree 79
- scripts
 - defined* 26
 - and objects 12
 - offline processing 24
 - online processing 24
 - proactive 12
 - reactive 12
 - resources 26
 - RMS Wizard Kit 24
 - timeout 255
 - top-of-tree 256

Index

- SCRIPTS_TIME_OUT 255
- ScriptTimeout attribute
 - defined* 239
- searching logs
 - in log viewer 171
- secondary management server, Web-Based Admin View 122
- secondary menus, Wizard Tools 99
- sending
 - clear-fault request 58
- setting
 - log levels with hvutil -L 269
 - log levels with Wizard Tools 270
- severity levels *see* error levels
- shutdown
 - RMS 181
 - status code 122
- Shutdown Facility
 - ensuring node shutdown 16
 - with RMS 13
- ShutdownPriority attribute
 - defined* 239
- SIS
 - tabbed view, Cluster Admin 126
- site preparation 215
- site preparation overview 215
- software monitor
 - function 1
 - RMS 10
- split-brain condition
 - display in clusterwide table 141
- SplitRequest attribute 243
- stale file handle
 - NFS 225
- Standby state
 - defined* 227
 - switching to 194
- StandbyCapable attribute 240
- starting
 - applications, automatically 188
 - applications, GUI 193
 - applications, hvswitch command 192
 - maintenance mode 208
 - RMS, and heartbeat recovery timeout 67, 177
 - RMS, automatically 179, 181

-
- RMS, GUI 173
 - RMS, hvcm command 176
 - startup file
 - RMS default 177
 - state machine, RMS base monitor 11
 - StateChangeScript
 - script executed at the time of state transition 27
 - StateDetails attribute
 - defined* 244
 - displaying 229
 - states
 - definitions* 227
 - and available operations 142
 - applications, controlling 188
 - changes, and initiating actions 11
 - changes, and scripts 12, 26
 - changes, confirmation of 133
 - controlling 173
 - Deact 227
 - details, displaying 229
 - determined by detectors 26
 - display, comparing 158
 - displayed in Cluster Admin GUI 135
 - displayed in clusterwide table 139
 - displayed in context menus 131, 153
 - displayed in RMS graph 11
 - displaying with CLI 136
 - Faulted 196, 227
 - Faulted, clearing 196
 - Inconsistent, *defined* 227
 - Inconsistent, in clusterwide table 141
 - intended, maintenance mode 210
 - Maintenance 228
 - names in clusterwide table 140
 - nodes, detected by ELM 178
 - nodes, detecting in RMS 11
 - objects, reported by detectors 12
 - Offline 227
 - OfflineFault 227
 - Online 227
 - online and offline processing 13
 - parent object, maintenance mode 208
 - reported by detectors 22
 - resources, monitored by RMS 11

Index

- RMS, and real world 14
- Standby 194, 227
- Unknown 227
- Wait 186, 228
- Warning 228
- state-triggered scripts
 - FaultScript 27
 - OfflineDoneScript 27
 - PostOfflineScript 27
 - PostOnlineScript 27
 - WarningScript 27
- status
 - codes, CLI *see* exit codes
 - icons *see* icons
- stopping
 - applications, GUI 194
 - applications, hvutil command 194
 - maintenance mode 211
 - RMS, GUI 181
 - RMS, hvshut command 185
- storage managers
 - administrator prerequisites 1
- strace, Linux trace tool 279
- subapplications
 - graph 149
 - in Cluster Admin RMS tree 129
- sub-menus, Wizard Tools 99
- summary, clusterwide table 139
- switching
 - defined* 60
 - applications 29, 285
 - applications, GUI 190
 - applications, hvswitch command 192
- switchlog
 - duplicate entries in system log 225
 - file 261, 262
 - location 34
 - script timeout error messages 60
 - viewing 162
- SysNode
 - switching application to 29
- SysNode objects
 - defined* 232
 - fault 59

- initializing 37
- overview 106
- representing nodes 13
- selecting 131
- switching application to 285
- Wait state, clearing 186
- system files, preparing for RMS 215
- system log
 - RMS messages 225

T

- tabbed view
 - Cluster Admin left pane 126
 - Cluster Admin msg 127
 - Cluster Admin rms 128
 - graphs 145, 164
 - log viewer 164
- tables
 - clusterwide 139
- timeouts
 - heartbeat recovery 65, 176
- trace
 - detector, set automatically 279
- truss, Solaris trace tool 279
- trusted
 - Java applets 125
 - login, required by Wizard Tools 217
- turnkey wizards
 - creating applications 95
 - DEMO 93
 - overview 91

U

- UDP protocol
 - RMS heartbeats 65, 249
- uname command 219
- Unknown state
 - defined* 227
 - exiting 37
 - initial state 37
 - RMS initialization 37
- us directory
 - RMS 33
- userApplication objects

Index

- defined* 232
- activating 198
- clearing faults 196
- context menus 131
- hvswitch command 192
- independent initialization 37
- overview 106
- representing applications 13
- resetting, GUI 195
- resetting, hvreset command 196
- RMS tree 129
- state information 139
- stopping, hvshut command 186
- taking offline 194

V

- /var/adm/messages 225
- variables
 - RMS environment 31
- /var/log/messages 225
- /var/opt/SMAWRrms/log/ 34, 162
- view
 - Cluster Admin initial 125
 - Cluster Admin message 127
- viewing
 - see also* displaying
 - application logs 163
 - composite subapplication graph 151
 - graphs, attaching and detaching 145, 146, 164, 165
 - GUI message 127
 - log viewer 164
 - object attributes 137
 - subapplication graphs 149
- virtual representation
 - RMS configuration 11
- volume managers
 - administrator prerequisites 1
 - as RMS resources 22

W

- Wait state
 - defined* 228
 - clearing faulted resources 285
 - clearing hung nodes 285

-
- clearing, SysNode 186
 - script timeout limit for node kill 60
 - Warning state
 - defined* 228
 - WarningScript 27, 240
 - warnings
 - log viewer 169
 - WarningScript
 - defined* 27
 - WarningScript attribute 27, 240
 - Web-Based Admin View
 - login 123
 - management server 122
 - Wizard Kit
 - configuration 90
 - RMS 23, 24
 - Wizard Tools
 - basic settings 100
 - configuring 90
 - DEMO turnkey 93
 - frequently used items 93
 - general description 90
 - hvexec command 105
 - main menu 94
 - non-basic settings 100
 - overview 90
 - resource wizards 91
 - secondary menus 99
 - setting detector log levels 270
 - sub-menus 99
 - turnkey 91, 95
 - Wizard Tools detectors
 - log file location 268
 - setting log level 270
 - wizards
 - debug level 271
 - debug reporting 270

Tables

Table 1: Available CLI commands	28
Table 2: RMS base directory structure	33
Table 3: Log directory structure	34
Table 4: Dependence of scalable controller states on child applications	75
Table 5: RMS severity level description	169
Table 6: RMS host name conventions in /etc/hosts	216
Table 7: States reported by detectors for RMS objects	227
Table 8: Additional states that may be displayed for RMS objects . . .	227
Table 9: StateDetails values for RMS objects	229
Table 10: Log files	262
Table 11: Base monitor log levels	265
Table 12: OID set in SNMP Trap	281

