

PRIMECLUSTER™

Reliant Monitor Services (RMS) with Wizard Tools (Solaris®, Linux®)
Configuration and Administration Guide

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion of this manual. Your feedback helps us optimize our documentation to suit your individual needs.

Fax forms for sending us your comments are included in the back of the manual.

There you will also find the addresses of the relevant User Documentation Department.

Certified documentation according DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Copyright © 2002, 2003, 2004, 2005, 2006, Fujitsu Siemens Computers Inc. and Fujitsu LIMITED.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

Sun, Sun Microsystems, Solaris, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

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

This manual is printed on
paper treated with
chlorine-free bleach.

Preface

Introduction

Using the Wizard Tools interface (hww)

Configuration example

Administration

Scalable controllers

Advanced RMS concepts

Appendix—Object types

Appendix—Attributes

Appendix—Environment variables

Continued ►

Appendix—List of manual pages

Glossary

Abbreviations

Figures

Tables

Index

Contents

1	Preface	1
1.1	About this manual	1
1.2	PRIMECLUSTER documentation list	2
1.3	Conventions	4
1.3.1	Notation	4
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	5
1.3.1.5	Example 1	5
1.3.1.6	Example 2	5
1.3.2	Command syntax	6
1.4	Important notes and cautions	6
1.5	Abbreviations	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	15
2.2.3	Node and application failover	16
2.2.4	Controlled applications and controller objects	17
2.2.4.1	Follow controllers	19
2.2.4.2	Scalable controllers	20
2.2.4.3	Further notes about controllers	21
2.3	How the Wizard Tools provide easy configuration	22
2.4	RMS wizard products	23
2.4.1	RMS Wizard Tools	25
2.4.2	RMS Wizard Kit	25
2.5	Cluster Admin administration tool	26
2.6	RMS components	26
2.6.1	Base monitor	26
2.6.2	Detectors and states	27
2.6.3	Scripts	29
2.6.4	RMS CLI	30
2.7	Object types	33
2.8	Object attributes	34
2.9	Environment variables	34
2.9.1	Script execution environment variables	35
2.10	RMS Directory structure	36

Contents

3	Using the Wizard Tools interface (hvw)	39
3.1	Overview	39
3.1.1	RMS Wizard types	40
3.1.1.1	Turnkey wizards	41
3.1.1.2	Resource wizards	41
3.2	Site preparation	42
3.2.1	Network database files	42
3.2.2	File systems—Solaris only	45
3.2.2.1	NFS Lock Failover—Solaris only	46
3.2.3	File systems—Linux only	47
3.2.4	Log files	48
3.2.5	Other system services and databases	48
3.3	General configuration procedure	49
3.4	Creating and editing a configuration	49
3.4.1	Using the wizard menus	50
3.4.2	Main configuration menu	51
3.4.2.1	Main configuration menu when RMS is not active	51
3.4.2.2	Main configuration menu when RMS is running	55
3.4.3	Secondary menus	56
3.4.4	Basic and non-basic settings	57
3.5	Activating a configuration	59
3.6	Configuration elements	63
3.6.1	Scripts	63
3.6.2	Detectors	64
3.6.3	RMS objects	64
3.7	Further reading	65
4	Configuration example	67
4.1	Stopping RMS	67
4.2	Creating a configuration	68
4.3	Adding hosts to the cluster	69
4.4	Creating an application	71
4.5	Entering Machines+Basics settings	74
4.6	Entering non-basic settings	78
4.7	Specifying a display	80
4.8	Activating the configuration	83
4.9	Creating a second application	85
4.10	Setting up a controlling application	89
4.11	Specifying controlled applications	90
4.12	Activating the configuration a second time	94
4.13	Starting RMS	95

5	Administration	97
5.1	Overview	97
5.2	Using Cluster Admin	98
5.2.1	Starting Cluster Admin	98
5.2.2	Logging in	99
5.2.3	Main Cluster Admin window	103
5.2.4	Cluster Admin message view	104
5.3	Viewing RMS status and attributes	105
5.3.1	RMS tree	106
5.3.2	Context-sensitive (pop-up) command menus	108
5.3.3	Confirmation pop-up windows	110
5.3.4	Displaying environment variables	110
5.3.5	Displaying object states	113
5.3.6	Configuration information or object attributes	114
5.3.7	Viewing RMS log messages	114
5.3.8	Using the RMS clusterwide table	123
5.3.8.1	Using context menus from the clusterwide table	126
5.3.9	Display during RMS configuration changes	126
5.4	Controlling RMS operation	128
5.4.1	Starting RMS	128
5.4.2	Starting RMS automatically at boot time	131
5.4.3	Stopping RMS	133
5.4.4	Overriding automatic application startup	137
5.4.5	Starting an application	140
5.4.6	Switching an application	141
5.4.7	Taking an application offline	142
5.4.8	Activating an application	143
5.4.9	Clearing a fault	144
5.4.10	Clearing a SysNode Wait state	145
5.4.11	Using maintenance mode	146
5.5	Related administrative procedures	154
5.6	Using RMS graphs	154
5.6.1	RMS full graph	154
5.6.2	Application graph	158
5.6.3	Subapplication graph	159
5.6.4	Composite subapplication graph	161
5.6.5	Using command pop-up menus from the graph	163
5.6.6	Changing the displayed detail level	164
5.6.7	Interpreting the graph after RMS shutdown	167

Contents

6	Scalable controllers	169
6.1	Controller overview	169
6.2	Scalable controllers and applications	169
6.2.1	Scalable Applications	170
6.2.2	Benefits of Scalable Controllers	170
6.2.3	Attributes for Scalable Controllers	171
6.3	Online/offline processing and state transitions	172
6.3.1	How controller states depend on controlled application states	172
6.3.2	Request propagation to controlled applications	173
6.3.3	Controller warning state	173
6.3.4	Application warning state	174
6.3.5	Controller state change script	174
6.3.6	Sequenced online/standby/offline and application groups	177
6.3.7	Auto Startup on a sub-cluster	178
6.3.8	Switchover on a sub-cluster	179
6.3.9	Manual switching of child applications	179
6.3.10	Operation during maintenance mode	179
6.3.10.1	Switching applications between nodes	180
6.3.10.2	Online, offline, and fault processing restrictions	182
6.3.10.3	Recommended approach for scalable controllers	183
7	Advanced RMS concepts	185
7.1	Internal organization	185
7.1.1	Application and resource description	185
7.1.2	Messages	186
7.2	Initializing	186
7.3	Online processing	191
7.3.1	Online request	191
7.3.1.1	Manual methods	192
7.3.1.2	Automatic methods	192
7.3.2	PreCheckScript	193
7.3.3	Online processing in a logical graph of a userApplication	194
7.3.4	Unexpected reports during online processing	196
7.3.5	Fault situations during online processing	197
7.3.6	Initialization when an application is already online	197
7.4	Offline processing	198
7.4.1	Offline request	198
7.4.2	Offline processing in a logical graph of a userApplication	199
7.4.3	Unexpected reports during offline processing	201
7.4.4	Fault situations during offline processing	201
7.4.5	Object is already in Offline state	202
7.4.6	Object cannot be sent to Offline state	202

7.5	Fault processing	202
7.5.1	Faults in the online state or request processing	203
7.5.2	Offline faults	205
7.5.3	AutoRecover attribute	205
7.5.4	Fault during offline processing	206
7.5.5	Examples of fault processing	206
7.5.6	Fault clearing	208
7.5.7	SysNode faults	210
7.5.7.1	Operator intervention	211
7.6	Switch processing	211
7.6.1	Switch request	211
7.7	Special states	213
7.7.1	Restrictions during maintenance mode	213
7.7.2	The Inconsistent state	214
8	Appendix—Object types	217
9	Appendix—Attributes	219
9.1	Attributes available to the user	219
9.2	Attributes managed by configuration wizards	226
10	Appendix—Environment variables	231
10.1	Setting environment variables	231
10.2	Global environment variables	232
10.3	Local environment variables	238
10.4	Script execution environment variables	242
11	Appendix—List of manual pages	245
11.1	CCBR	245
11.2	CF	245
11.3	CFS	246
11.4	CIP	246
11.5	Monitoring Agent	247
11.6	PAS	247
11.7	RCVM	248
11.8	Resource Database	248
11.9	RMS	249
11.10	RMS Wizards	251
11.11	SCON	251
11.12	SF	251
11.13	SIS	252
11.14	Web-Based Admin View	252

Contents

Glossary 255

Abbreviations 273

Figures 277

Tables 285

Index 287

1 Preface

PRIMECLUSTER™ 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.

The 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
- 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 the *PRIMECLUSTER Installation Guide* for your operating system.

1.1 About this manual

This manual is structured as follows:

- The chapter “Introduction” on page 9 provides an introduction to RMS terminology and describes basic principles of operation.
- The chapter “Using the Wizard Tools interface (hvw)” on page 39 describes how to configure RMS using the RMS Wizard Tools.
- The chapter “Configuration example” on page 67 illustrates the Wizard Tools configuration process for two simple applications on a small cluster.
- The chapter “Administration” on page 97 discusses how to administer RMS with the Cluster Admin GUI, including the equivalent CLI procedure for some functions.
- The chapter “Scalable controllers” on page 169 describes operational details of scalable controllers and scalable applications.
- The chapter “Advanced RMS concepts” on page 185 provides details about state detection and transition processing.

- The chapter “Appendix—Object types” on page 217 lists the object types that are supplied with RMS.
- The chapter “Appendix—Attributes” on page 219 lists the attributes that are supported by RMS object types.
- The chapter “Appendix—Environment variables” on page 231 describes the RMS environment variables.
- The chapter “Appendix—List of manual pages” on page 245 lists the manual pages for PRIMECLUSTER.

1.2 PRIMECLUSTER documentation list

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

- *Concepts Guide (Solaris, Linux)*—Provides conceptual details on the PRIMECLUSTER family of products.
- *Installation Guide (Solaris)*—Provides instructions for installing and upgrading PRIMECLUSTER products.
- *Installation Guide (Linux)*—Provides instructions for installing and upgrading PRIMECLUSTER products.
- *Web-Based Admin View (Solaris) Operation Guide*—Provides information on using the Web-Based Admin View management GUI.
- *Web-Based Admin View (Linux) Operation Guide*—Provides information on using the Web-Based Admin View management GUI.
- *Cluster Foundation (CF) (Solaris) Configuration and Administration Guide*—Provides instructions for configuring and administering the PRIMECLUSTER Cluster Foundation.
- *Cluster Foundation (CF) Configuration and Administration Guide (Linux)*—Provides instructions for configuring and administering the PRIMECLUSTER Cluster Foundation.
- *Reliant Monitor Services (RMS) with Wizard Tools (Solaris, Linux) Configuration and Administration Guide*—Provides instructions for configuring and administering PRIMECLUSTER Reliant Monitor Services using the Wizard Tools interface.

- *Reliant Monitor Services (RMS) with PCS (Solaris, Linux) Configuration and Administration Guide*—Provides instructions for configuring and administering PRIMECLUSTER Reliant Monitor Services using the PCS (PRIMECLUSTER Configuration Services) interface.
- *Reliant Monitor Services (RMS) (Solaris, Linux) Troubleshooting Guide*—Describes diagnostic procedures to solve RMS configuration problems, including how to view and interpret RMS log files. Provides a list of all RMS error messages with a probable cause and suggested action for each condition.
- *Scalable Internet Services (SIS) (Solaris, Linux) Configuration and Administration Guide*—Provides information on configuring and administering Scalable Internet Services (SIS).
- *Global Disk Services (Solaris) Configuration and Administration Guide*—Provides information on configuring and administering Global Disk Services (GDS).
- *Global File Services (Solaris) Configuration and Administration Guide*—Provides information on configuring and administering Global File Services (GFS).
- *Global Link Services (Solaris) Configuration and Administration Guide: Redundant Line Control Function*—Provides information on configuring and administering the redundant line control function for Global Link Services (GLS).
- *Global Link Services (Solaris) Configuration and Administration Guide: Multipath Function*—Provides information on configuring and administering the multipath function for Global Link Services (GLS).
- *Data Management Tools (Solaris) Configuration and Administration Guide*—Provides reference information on the Volume Manager (RCVM) and File Share (RCFS) products. (Not available in all markets)
- *SNMP Reference Manual (Solaris, Linux)*—Provides reference information on the Simple Network Management Protocol (SNMP) product.
- Release notices for all products—These documentation files are included as HTML files on the PRIMECLUSTER Framework CD. Release notices provide late-breaking information about installation, configuration, and operations for PRIMECLUSTER. Read this information first.
- *RMS Wizards documentation package*—Available on the PRIMECLUSTER CD. These documents deal with Wizard Tools topics such as the configuration of file systems and IP addresses. They also describe the various types of available RMS wizards.

- PCS Migration Guide—Available on the PRIMECLUSTER CD. Describes procedures for converting configurations from the Wizard Tools environment to PCS.

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 `fuj i2#` would mean that the command was run as user `root` on the node named `fuj i2`.

1.3.1.2 Manual page section numbers

References to operating system commands are followed by their manual page section numbers in parentheses—for example, `cp(1)`.

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, manual page 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, e.g., <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 syntax

The command 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

Material of particular interest is preceded by the following symbols in this manual:



Contains important information about the subject at hand.



Caution

Indicates a situation that can cause harm to data.

1.5 Abbreviations

- Solaris™ operating system is abbreviated as Solaris.
- If "Solaris X" is indicated in the reference manual name of the Solaris manual, replace "Solaris X" with "Solaris 9 operating system (Solaris 9)" or the "Solaris 10 operating system (Solaris 10)".

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.

This chapter contains the following sections:

- “PRIMECLUSTER overview” on page 9
- “How RMS provides high availability” on page 11
- “How the Wizard Tools provide easy configuration” on page 22
- “RMS wizard products” on page 23
- “Cluster Admin administration tool” on page 26
- “RMS components” on page 26
- “RMS CLI” on page 30
- “Object types” on page 33
- “Object attributes” on page 34
- “Environment variables” on page 34
- “RMS Directory structure” on page 36

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.

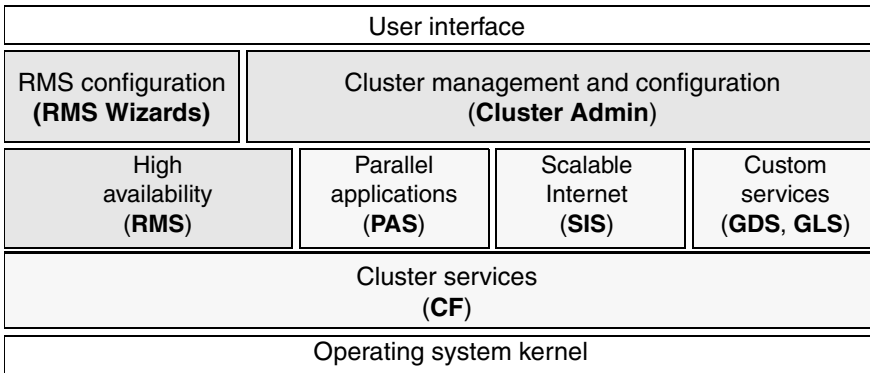


Figure 1: Overview of PRIMECLUSTER products

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. See your sales representative 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.

Other PRIMECLUSTER products (shown with a dotted background in the figure above) are described in their respective manuals. See the section “Related documentation” 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. RMS also monitors the state of each host in the cluster.

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.

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 to detect these problems.

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. 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.

In addition, 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 go 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. In this case, when RMS starts up, the expert must also manually adjust the heartbeat recovery timeout to a smaller value.

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**.

Machines that are members of a cluster are called **nodes**. Each node that may run one or more applications in the high availability configuration is represented by an RMS `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.

A complete list of the RMS object types supported by the Wizards Tools is presented in the chapter "Appendix—Object types" on page 217.

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` from the offline `SysNode` 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 the *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.

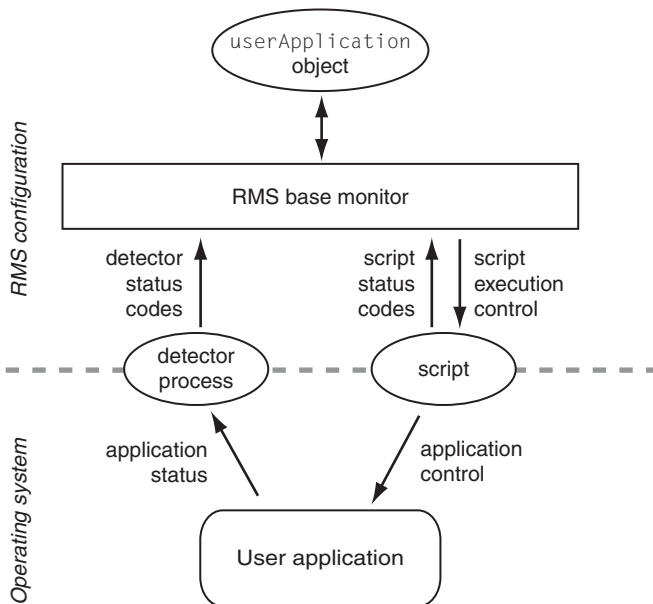


Figure 2: Interface between RMS and the operating system

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, it is customary to say, “the `xyz` file system is offline,” rather than the strictly correct but more verbose, “the `gResource` object named `xyz` is in the `Offline` state.” It is also common to refer to a script by its attribute name, e.g., “the script specified by the `PreOnlineScript` attribute” becomes simply “the `PreOnlineScript`.”

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.

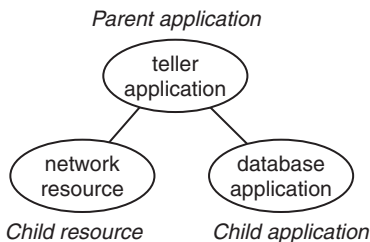


Figure 3: Parent application with two dependencies

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**. Like resource objects, a controller is configured with detectors and scripts: the detectors monitor the state of the child (controlled) application, and the scripts implement appropriate responses by the parent (controlling) 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.

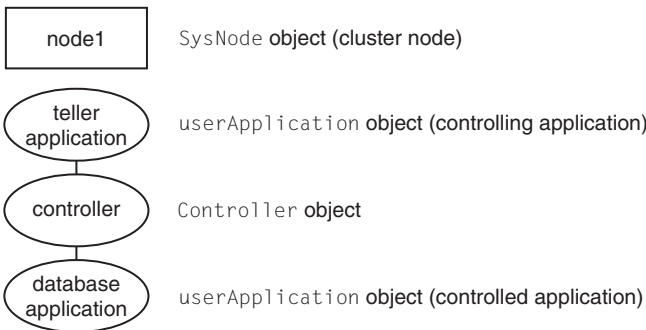


Figure 4: RMS representation of controlled application

i 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.

Failover of controlled applications

If a child changes to an offline or faulted state, RMS will attempt to switch the parent, the child, and/or the dependent resources to another node. other nodes. The exact action depends on whether the controller has been configured to operate in **follow** or **scalable** mode, as discussed below.

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 the 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.

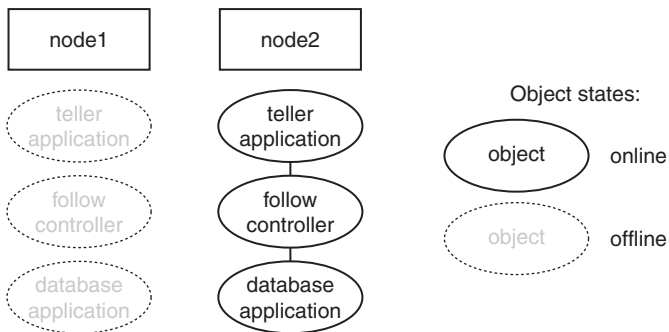


Figure 5: Result of follow mode switchover

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.

i 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 complete discussion, see the chapter “Advanced RMS concepts” on page 185.

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.

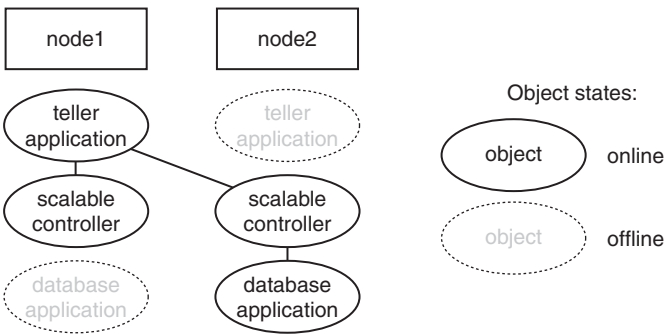


Figure 6: Scalable mode child (controlled) application switchover

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.

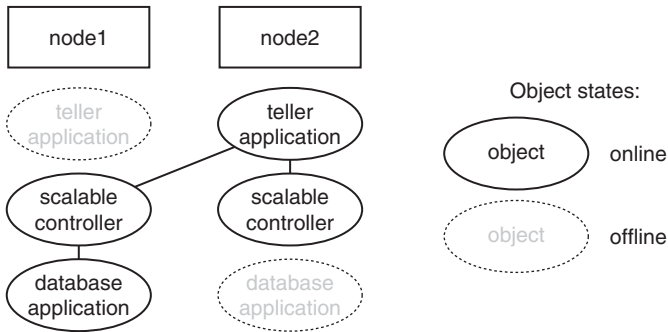


Figure 7: Scalable mode parent (controlling) application switchover

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.

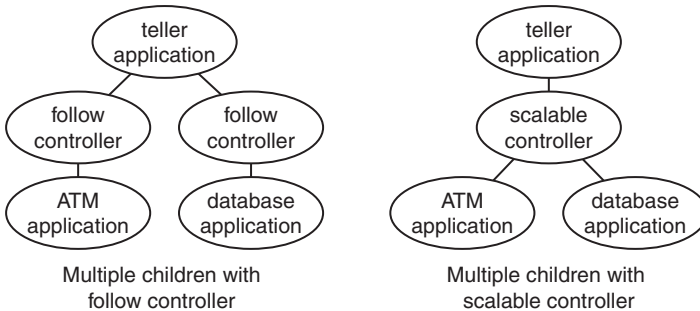


Figure 8: Controller structure with multiple children

For a more detailed discussion, see the chapter “Scalable controllers” on page 169.

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 wizards 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 wizards 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 or SAP R/3, 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.

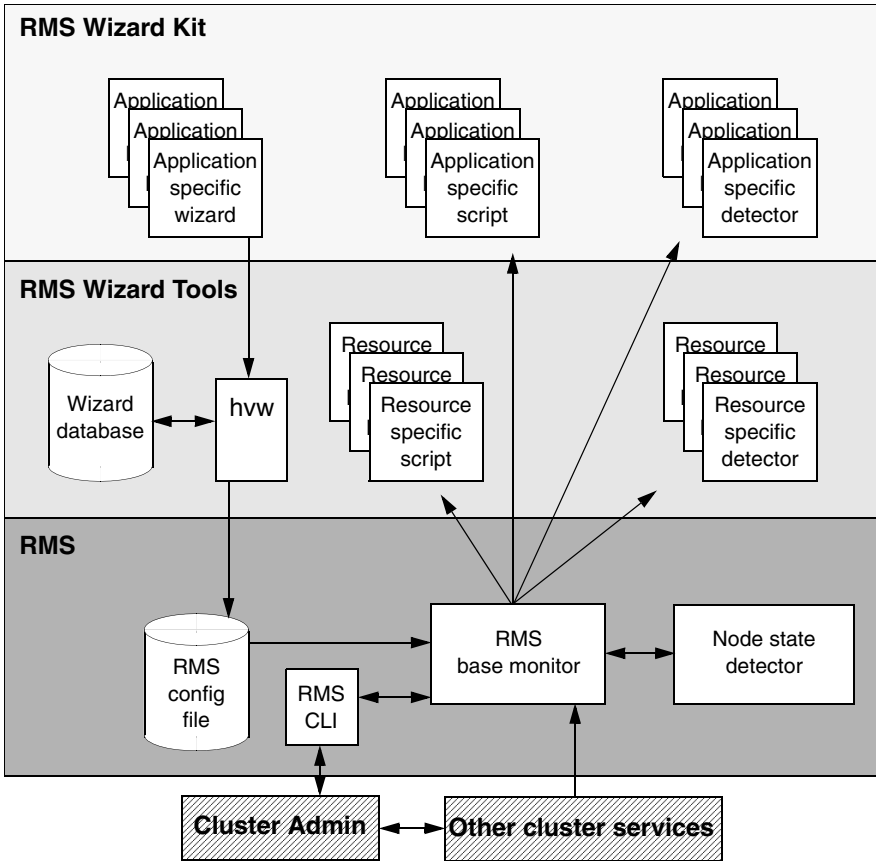


Figure 9: Relationship between RMS and RMS Wizards

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 or refer to the RMS Wizards documentation package.

2.5 Cluster Admin administration tool

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

- Application startup
- Application shutdown
- Manual application switchover
- Visual cues for resource and application fault isolation
- Fault clearing capability
- RMS startup
- RMS shutdown
- Graphs of application and resources

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

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 depicted by objects, their attributes, and their interdependent relationships
- Receives requests from the RMS command line interface (CLI) to take actions
- Monitors the heartbeat from every node to keep track of each machine's status and its connectivity to the rest of the cluster

- Receives input from detectors that report 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 as required by a CLI request or in case of a resource or node failure
- Performs various administrative functions

2.6.2 Detectors and states

Detectors are independent processes that monitor specific sets of resources in order to determine their state. The detector does not determine if the current state of a resource is the correct state or not (for example, if a resource is `Offline` but is supposed to be `Online`)—that is the role of the base monitor.

Detectors can report the following states to the base monitor:

<code>Faulted</code>	Error condition encountered. The error may have occurred in the resource, in one of its children, or during script processing.
<code>Offline</code>	Disabled, not ready for use. The scripts have successfully disabled the resource.
<code>Online</code>	Enabled, ready for use. All required children are online, and no errors were encountered while scripts were processed.
<code>Standby</code>	Ready to be quickly brought <code>Online</code> when needed.

The following resource states may also be displayed in the GUI status area:

<code>Deact</code>	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> .
<code>OfflineFault</code>	Fault that occurred in the past has not yet been cleared.
Unknown	No information is available. Reported before object initialization is completed.
<code>Wait</code>	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.
Maintenance	<p>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.</p> <p>An application in maintenance mode is usually marked with its intended state, which is the state that would be attained if the application were immediately taken out of maintenance mode. The maintenance mode intended states are <code>Maintenance-Online</code>, <code>Maintenance-Offline</code>, and <code>Maintenance-Standby</code>.</p>

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`.

Detectors for common system functions are provided by the Wizard Tools. Additional application-specific detectors are included with the Wizard Kit.

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`—Runs only once when RMS is first started
- `PreCheckScript`—Determines if `Online` or `Standby` processing is needed or possible
- `PreOfflineScript`—Prepares a transition to an `Offline` state
- `OfflineScript`—Transitions a resource to an `Offline` state
- `PreOnlineScript`—Prepares a transition to an `Online` state
- `OnlineScript`—Transitions a resource to an `Online` state

- State-triggered scripts react to specific events.

The state-triggered scripts are as follows:

- `PostOnlineScript`—Reaction to the transition to the `Online` state
- `PostOfflineScript`—Reaction to the transition to the `Offline` state
- `OfflineDoneScript`—Reaction to a `userApplication` reaching the `Offline` state
- `FaultScript`—Reaction to a resource transitioning to the `Faulted` state
- `WarningScript`—Reaction to a detector reporting the `Warning` state

Scripts for common system functions are included with the subapplications provided by the 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, and, under certain conditions, you may find it useful to invoke the CLI directly.

Table 1 lists the RMS CLI commands available to administrators. Specific procedures using some of these commands are described in the chapter “Administration” on page 97. For a complete description of any command’s usage, see its online `man` page. For a list of all commands related to RMS, see the chapter “Appendix—List of manual pages” on page 245.



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

Command	Function
<code>hvassert</code>	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.
<code>hvattrib</code>	Provides an interface for changing the <code>AutoSwitchOver</code> attribute at runtime. The change can be made from a single node in the cluster and will be applied clusterwide for one or more <code>userApplication</code> objects in the currently running configuration. The values <code>No</code> , <code>HostFailure</code> , <code>ResourceFailure</code> , or <code>ShutDown</code> may be specified. <code>hvattrib</code> command arguments are specific to objects and attributes in RMS configurations. The user should be familiar with these attributes, which are described in the chapter “Appendix—Attributes” on page 219.

Table 1: Available CLI commands

Command	Function
hvcm	<p>Starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the <code>hvcm</code> 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 <code>hvconfig</code> 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.
hvdist	Distributes the configuration file to all nodes within an RMS configuration.
hvdump	Gets debugging information about RMS on the local node.
hvgdmake	Makes (compiles) a custom detector so that it can be used in the RMS configuration. The user first prepares a source file for the detector, which must be a file with a <code>.c</code> extension.
hvlogclean	Either saves old log files into a subdirectory whose name is the time RMS was last started, or, if invoked with the <code>-d</code> option, deletes old log files. In either case, <code>hvlogclean</code> creates a clean set of log files even while RMS is running.

Table 1: Available CLI commands

Command	Function
hvrcllev	Displays or changes the run level used for RMS when it is started automatically at system startup. During installation, pkgadd uses hvrcllev to set the RMS run level to the default in /etc/inittab. If the system default run level is changed at a later time, the RMS run level should be adjusted accordingly with hvrcllev to ensure that RMS starts in the proper sequence. hvrcllev may also be used to display the current system default run level.
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. This command is intended for use by experts only.
hvsetenv	Provides an interface for changing the following RMS environment variables on the local node: <ul style="list-style-type: none"> – 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” on page 231.
hvshut	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 user application resource from one system node to another in the RMS configuration. The resource being switched must be of type userApplication. The system node must be of type SysNode.

Table 1: Available CLI commands

Command	Function
hvthrottle	Prevents multiple scripts within a configuration file from running at the same time by creating queues for sequential processing.
hvutil	Provides general administration interface to RMS. It performs various resource administration tasks, such as dynamically setting logging levels, sending a resource <code>Offline</code> , clearing faulted resources or hung cluster nodes in the <code>Wait</code> state, setting detector time periods, setting Maintenance Mode, and so forth.

Table 1: Available CLI commands

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`
- `andOp`
- `orOp`
- `Controller`

Refer to the chapter “Appendix—Object types” on page 217 for the supported types, their required attributes, and a 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 chapter “Appendix—Attributes” on page 219 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 chapter “Using the Wizard Tools interface (hvw)” on page 39.

2.9 Environment variables

RMS uses global and local environment variables:

- Global variables generally control clusterwide operations and must have the same setting on all nodes in the cluster. At runtime, RMS maintains global environment variables in the `ENV` object.



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 variables can differ from node to node. RMS maintains local environment variables in the `ENVL` object.

RMS creates the `ENV` and `ENVL` objects dynamically when the base monitor starts up:

1. First, it loads global variables from the `<RELIANT_PATH>/bin/hvenv` file, which is installed with the package.



Caution

Do not modify the `<RELIANT_PATH>/bin/hvenv` file.

- Next, it loads both global and local variables from the `<RELIANT_PATH>/bin/hvenv.local` file, which contains configuration-specific variables that are typically set by the Wizard Tools. Experts may change the contents of this file manually with a standard text editor. In any case, changes to the `hvenv.local` file will not take effect until the next RMS startup.

i The `RELIANT_PATH` global variable is defined at installation. By default, it is set to `/opt/SMAW/SMAWRrms`.

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

While RMS is running, you can display the environment variables with the `hvdisp` command, which does not require root privilege:

- Use `'hvdisp ENV'` to display the the global list.
- Use `'hvdisp ENVL'` to display the local list.

Refer to the chapter “Appendix—Environment variables” on page 231 for a description of all global and local environment variables. The appendix also describes how to change the value of any environment variable.

2.9.1 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” on page 242 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/SAW/SAWRrms</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. The base monitor and detectors create log files here. Default: <code>/var/opt/SMAWRrms/log</code></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 <code>yyyy-mm-dd_HH:MM:SS</code> to indicate the date and time when the backup was created.</p>

Table 3: Log directory structure

3 Using the Wizard Tools interface (hvw)

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

- The section “Overview” on page 39 gives a brief overall description of the configuration process and the RMS Wizards.
- The section “General configuration procedure” on page 49 outlines the four major steps involved in every configuration procedure.
- The section “Creating and editing a configuration” on page 49 describes the wizard interface and how it is used to specify a configuration.
- The section “Activating a configuration” on page 59 describes how to activate a configuration after it has been created or modified.
- The section “Configuration elements” on page 63 provides additional details about basic RMS elements specified in every configuration.
- The section “Further reading” on page 65 contains a list of related documents that provide additional information about the wizards.

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

3.1 Overview

The chapter “Introduction” on page 9 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.

3.1.1 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
 - The GENERIC application wizard, which allows you to configure a wide range of applications
 - The DEMO wizard, which provides a simple demonstration of the Wizard Tools and RMS
 - The resource wizards-oriented wizards, which provide scripts and detectors for basic resources such as file systems, volume managers, and IP addresses. They are used by the GENERIC and DEMO 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 R/3 and ORACLE wizards are components of the Wizard Kit.



For information on the availability of the RMS Wizard Kit, contact your local customer support service or refer to the RMS Wizards documentation package. See the section “Further reading” on page 65 for more information.

3.1.1.1 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 are designed to configure a specific type of application. The configuration described in the chapter “Configuration example” on page 67 uses the GENERIC and DEMO turnkey wizards. Other examples are the R/3 wizard and the ORACLE wizard. By convention, turnkey wizards have names with all uppercase letters.

3.1.1.2 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 Global Disk Services (GDS).
- **Gls**—Configures the IP addresses administrated by Global Link Services (GLS).
- **Ippaddress**—Configures the IP addresses that are needed for communication over a LAN interface.
- **Rcvm**—Configures disk groups administrated by the PRIMECLUSTER Volume Manager (not available in all areas).
- **Vxvm**—Configures disk groups administrated by the Veritas volume manager (not available in all areas).

3.2 Site preparation

The *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.

3.2.1 Network database files

- `/etc/hosts`

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 4.

Entry type	RMS naming pattern	Examples
Primary host name	<code><hostname>RMS</code>	fuji2RMS fuji3RMS
Alternate interfaces (AlternateIps)	<code><hostname>rmsAI<nn></code> where <code><nn></code> is a zero-filled sequence number in the range 01 to 99	fuji2rmsAI01 fuji2rmsAI02

Table 4: RMS host name conventions in `/etc/hosts`



The primary 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.

Example

The following entries are for a cluster with hosts `fuji2` and `fuji3`, each of which have two alternate network interfaces on different subnets:

```
172.25.219.83 fuji2
172.25.219.84 fuji3
# host names for RMS
192.168.1.1 fuji2RMS
192.168.1.2 fuji3RMS
192.168.2.11 fuji2rmsAI01 # fuji2 subnet 2 alt
192.168.2.12 fuji3rmsAI01 # fuji3 subnet 2 alt
192.168.3.21 fuji2rmsAI02 # fuji2 subnet 3 alt
192.168.3.22 fuji3rmsAI02 # fuji3 subnet 3 alt
```

RMS, like other PRIMECLUSTER products, also supports IPV6 addresses.

- `/.rhosts` (Solaris) and `/root/.rhosts` (Linux)

Contains entries to control trusted login from remote hosts.

The Wizard Tools require automatic login as `root` on every machine in the cluster, so the `.rhosts` file 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 `fujii2` and `fujii3`, then every machine's `.rhosts` file should contain the following lines:

```
fujii2 root
fujii3 root
```



The Cluster Foundation (CF) provides the equivalent of the `.rhosts` functionality for all RMS configuration, administration, and operation tasks.

- `/opt/SMAW/SMAWRrms/etc/hvipalias`

Contains entries for all of the LAN interfaces that are to be used as resources in the configuration. Typically, each alias associates a fixed IP address or host name with an interface on the local node. The alias always presents the same address or host name, even though it is switched from node to node, and even if the underlying physical interface has different characteristics on each node.

The entries must provide the names and netmasks that are required for the LAN. Optionally, there may also be some routing information. See the online document `Ipaddress.htm` or the header of the `hvipalias` file for the format of the entries.

Example

```
#uname  IfName  Interface  Netmask
fujii2  dbhost  eth1      0xffffffff00
```

An `hvipalias` entry may also contain the optional keyword `IFCONFIG` and a set of arguments to be sent to the `ifconfig` command. This allows you to specify custom interface settings that may be required for the physical device.

The `IFCONFIG` keyword must be specified as all uppercase letters, followed by whitespace, followed by the comma-delimited argument string that will be passed to the `ifconfig` command. The keyword and arguments must appear after the `Netmask` field and before the `Routes` arguments.

For example, if an `mtu` value of 1200 is required for the local interface associated with the `dbhost` alias, the entry would be as follows:

```
#uname  IfName  Interface  Netmask
fujii2  dbhost  eth1      0xffffffff00  IFCONFIG mtu,1200
0
```


After an alias with IFCONFIG arguments is successfully brought online, the `ifconfig` command with the IFCONFIG arguments will be invoked for the associated device.

- `/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 `hvconsoles` online manual or in the comments in the `hvconsoles` file.

Example

```
ANY fuji2 echo GENERAL_ALERT_ARG
```

3.2.2 File systems—Solaris only

- `/etc/vfstab`

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`

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=
fuji2RMS:fuji2:045nfs045dial:045msg:fuji2RMS: /sapmnt/045
```

3.2.2.1 NFS Lock Failover—Solaris only

NFS Lock Failover feature applies to local file systems. If you enable NFS Lock Failover and the file system subsequently fails, the NFS locks associated with the file system also fail over along with the file system. To take advantage of this feature, the following site preparation steps need to be taken:

- You must have a shared disk accessible to all nodes in the cluster.
- Internal implementation of NFS Lock Failover needs a dedicated directory. You need to specify a directory that does not already exist. The directory will be used solely for NFS Lock Failover. Therefore, if you specify a directory that already exists, no other applications will be allowed to use it thereafter.

From the RMS *Main configuration menu*, select *Configuration-Edit-Global-Settings*. In the *Global Settings* menu, select menu item *NFSLockFailover* (see Figure 1). The directory entered in this screen will be created on all shared file systems selected for NFS Lock Failover. For example, if the directory `nfs_lock_dir` is entered in this screen and the file system `/usr/test1` in userApplication APP1 is selected for NFS Lock Failover, then a directory `/usr/test1/nfs_lock_dir` will be created (if it does not already exist) and will be used for storing lock information.



Only one file system per `userApplication` object can be selected for NFS Lock Failover. For a more detailed description, refer to the HTML documentation for the `Fsystem` wizard.

```
Shared Directory for NFS Lock Failover: Currently set -
1) HELP
2) FREECHOICE
3) RETURN
Global setting: Enable NFS Lock Failover:
```

Figure 10: NFS Lock Failover screen

- The directory entered in this screen must be accessible to all the nodes in the cluster. Otherwise, NFS failover will not work.
- This directory is reserved for NFS Lock Failover only.



This directory must not be used by any other applications.

- If the directory entered by the user begins with a slash (/) character, this character is dropped before creating the `/usr/test1/<nfs_lock_dir>` directory.
- Reserve one IP address for each `userApplication` object from which all the local file systems (set with NFS Lock Failover) must be shared.

3.2.3 File systems—Linux only

- `/etc/fstab`

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 need to be mounted locally.

For each file system to be managed by RMS, create a line 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.

Example

```
#RMS#/dev/sdb2 /fs2          ext2    defaults 1 2
```

- `/etc/exports`

Contains entries for all file systems that are available for mounting on other hosts.

For each file system to be managed by RMS, create a line with the standard `exports` fields, and then insert the string `#RMS#` at the beginning of the line. For more information, see the `exports` manual page.

Example

```
#RMS#/usr    fuji*(rw)
```

3.2.4 Log files

- `/var/adm/messages` (Solaris) or `/var/log/messages` (Linux)

By default, all RMS messages go to both the system log, `messages`, and the RMS `switchlog` file. If you do not want to send messages to the system log, then set `HV_SYSLOG_USE = 0` in the `hvenv.local` file. The default is `1`.



The backup files, `messages.0`, `messages.1`, etc., may also be useful for troubleshooting.

3.2.5 Other system services and databases

RMS requires the following system services or databases to be configured according to the instructions in the *PRIMECLUSTER Installation Guide* for your operating system:

- PRIMECLUSTER Cluster Foundation (CF), including CIP
- `/etc/nsswitch.conf` system service lookup order database
- `echo` service—Linux only

3.3 General configuration procedure

RMS configuration always involves these four steps:

▶ **Stop RMS.**

Refer to the section “Stopping RMS” on page 133. You can use the Cluster Admin GUI or the command line interface from any node in the cluster.

▶ **Create or edit the configuration.**

The next section provides general information, and the chapter “Configuration example” on page 67 walks through an example.

▶ **Activate the configuration.**

Activation includes generation and distribution. See the section “Activating a configuration” on page 59.

▶ **Start RMS.**

Refer to the section “Starting RMS” on page 128. 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 the *PRIMECLUSTER Installation Guide* for your operating system.

3.4 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 `mydemo` using the DEMO turnkey wizard. This example would be called up as follows:

```
hvw -n mydemo
```

The `hvw` command is documented in the online manual pages. Refer to the chapter “Appendix—List of manual pages” on page 245 for additional information.

3.4.1 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.

3.4.2 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.

3.4.2.1 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 11 shows the *Main configuration menu* window when RMS is inactive.

```
fuji2: Main configuration menu, current configuration: mydemo
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:
```

Figure 11: 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 (only for expert users)—Non-turnkey mode is meant for advanced/expert users only. If this mode is to be used, some rules must be followed. Otherwise, the resulting configuration may remain in an inconsistent state and RMS will not start. Usage of this mode is not within the scope of this guide.

- *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.
- *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. If one or more nodes is not available during the distribution phase, you can later use *Configuration-Push* (described in the next section) to update only those nodes.



Configuration-Activate is not available if RMS is already running on one or more nodes.

- *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.

Creating and editing a configuration Using the Wizard Tools interface (hvw)

- *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.

- *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.

- *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.

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.

- *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.

3.4.2.2 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 12.

```
fuji2: Main configuration menu, current configuration: mydemo
RMS up on: fuji2RMS -- RMS down on: fuji3RMS
1) HELP
2) QUIT
3) Application-View
4) Configuration-Generate
5) Configuration-Copy
6) Configuration-Remove
7) Configuration-Freeze
8) Configuration-Edit-Global-Settings
9) Configuration-Consistency-Report
10) Configuration-ScriptExecution
11) Configuration-Push
12) RMS-ViewMachine
Choose an action:
```

Figure 12: 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.

Creating and editing a configuration Using the Wizard Tools interface (hvw)

- *Configuration-Remove*—Removes (deletes) any existing configuration except the one that is currently running.
- *Configuration-Push*—Distributes a complete copy of the currently running configuration to a specific cluster node.

When a configuration is activated, some nodes may not be available. This menu item allows you to update individual cluster nodes that are brought up later, when RMS is already running. For example, if you changed the configuration while a node was down for maintenance, you could use *Configuration-Push* to update the node after it was restarted.



Configuration-Push is available only after the configuration has been activated.

- *RMS-ViewMachine*—Displays the list of nodes on which RMS is currently running.

3.4.3 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 13) is an example of a secondary menu. You see this menu after selecting *Application-Crete* from the main menu.

```
Creation: Application type selection menu:
1) HELP
2) QUIT
3) RETURN
4) OPTIONS
5) DEMO
6) GENERIC
7) LIVECACHE
8) R3ANY
9) R3CI
10) RTP
Application Type: 5
```

Figure 13: Application type selection



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

This option allows you to select an application type to be assigned to the application in question. This is an important step in the configuration procedure since it invokes the specific application-type wizard to provide all the predefined elements (for example, scripts and detectors) that go with that application type.

The chapter “Configuration example” on page 67 shows how to use some of the secondary menus. A more detailed description of these menus is given in the RMS Wizards documentation package.

3.4.4 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 5) *DEMO* produces the menu in Figure 14.

```
Consistency check ...
Yet to do: process the basic settings using Machines+Basics
Yet to do: choose a proper application name

Settings of turnkey wizard "DEMO"
1) HELP
2) NO-SAVE+EXIT
3) SAVE+EXIT
4) REMOVE+EXIT
5) ApplicationName=APP3
6) BeingControlled=no
7) Machines+Basics(-)
Choose the setting to process: 7
```

Figure 14: Menu leading to basic settings

If you select 7) *Machines+Basics*, you can configure the basic settings using the menu in Figure 15.

Creating and editing a configuration Using the Wizard Tools interface (hvw)

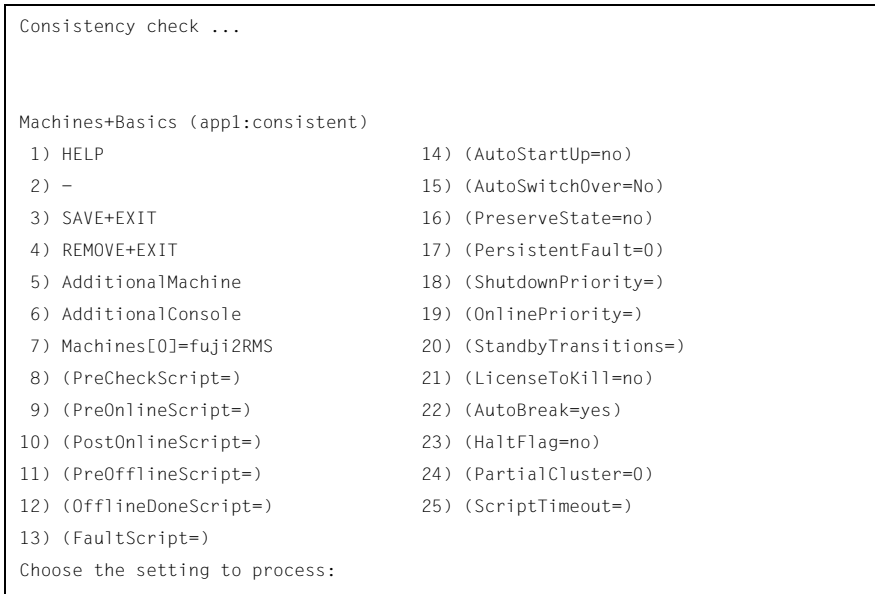


Figure 15: Menu to configure basic settings

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 16). 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 "DEMO"
1) HELP                                11) RemoteFileSystems(-)
2) -                                    12) IpAddresses(-)
3) SAVE+EXIT                            13) RawDisks(-)
4) -                                    14) RC-VolumeManagement(-)
5) ApplicationName=APP1                 15) VERITAS-VolumeManagement(-)
6) Machines+Basics(app1)               16) EMC-RdfManagement(-)
7) CommandLines(-)                    17) FibreCat-MirrorView(-)
8) Controllers(-)                     18) Gds:Global-Disk-Services(-)
9) DEMO(-)                             19) Gls:Global-Link-Services(-)
10) LocalFileSystems(-)
Choose the setting to process:

```

Figure 16: 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.

3.5 Activating a configuration

As described in section “General configuration procedure” on page 49, 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 nodes in the cluster before you activate a configuration.

The starting point for the activation phase is the *Main configuration menu* (see Figure 17).

```

fuji2: Main configuration menu, current configuration: mydemo
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 17: 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 18).


```

About to activate the configuration mydemo ...

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/mydemo/mydemo.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 18: 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 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.



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 19).

```
fuji2: Main configuration menu, current configuration: mydemo
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 19: 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
```

3.6 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.

i 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.

3.6.1 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 RMS Wizards provide a complete set of scripts for several pre-defined application types such as R/3 or Oracle. If you assign your application to one of these standard types, you automatically take advantage of the built-in scripts.

i 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, which is described in the section “Further reading” on page 65.

3.6.2 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.

Like the built-in scripts described in the previous section, the RMS Wizards provide built-in detectors for pre-defined application types. If you assign your application to one of these standard types, it automatically uses the built-in detectors.

3.6.3 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.

3.7 Further reading

The preceding sections were intended to make the reader familiar with some basic concepts and methods of the RMS Wizards. More information may be obtained from a number of documents that provide further reading on these tools and the way they are used.

RMS Wizards documentation package

The RMS Wizards documentation package is available in HTML format on the PRIMECLUSTER CD-ROM. After installation, the documents can also be found in the following locations:

`/opt/SMAW/SMAWRrms/htdocs.solaris/wizards.en/` (Solaris)

`/usr/doc/packages/SMAWRhv-do/wizards.en/` (Linux)

The information is presented in the following files and subdirectories:

- `feature_description.htm`

Describes the features added to recent versions of the RMS Wizard Tools.
- `primer.htm`

Provides an introduction to the RMS Wizards, covering many features in more detail than is possible in this chapter.
- `wizards/`

Provides information on individual wizards of all three kinds described in this chapter. Covers turnkey wizards, resource wizards, and other wizards, including the `generic` wizard.
- `scripts_and_tools/`

Provides information on some scripts and tools that may be useful in setting up a high-availability configuration by means of the RMS Wizards. Includes `gresources.htm`, which contains descriptions of a number of detectors.
- `manuals/`

Provides current manual pages for commands that are frequently used to configure an application with the RMS Wizards. The `hvw` and the `hvexec` commands, which were also described in this chapter, are explained here in more detail.

Manual pages

Information on the commands that are used for configuration with the RMS Wizards may also be obtained by calling up the online manual pages with the `man` command.

Manual pages are available, for instance, for the `hvw`, `hvcn`, and `hvexec` commands, which were used in the procedures described in this chapter.

4 Configuration example

This chapter provides an example of the configuration process using the RMS Wizards. Two simple applications are configured for operation on a small cluster. The example includes the following steps:

- “Stopping RMS” on page 67
- “Creating a configuration” on page 68
- “Adding hosts to the cluster” on page 69
- “Creating an application” on page 71
- “Entering Machines+Basics settings” on page 74
- “Entering non-basic settings” on page 78
- “Specifying a display” on page 80
- “Activating the configuration” on page 83
- “Creating a second application” on page 85
- “Setting up a controlling application” on page 89
- “Specifying controlled applications” on page 90
- “Activating the configuration a second time” on page 94
- “Starting RMS” on page 95

An abbreviated version of this example appears in the *PRIMECLUSTER Installation Guide* for your operating system.



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

4.1 Stopping RMS

Before you create or edit a configuration, ensure that RMS is not active on any machine that would be affected by the changes. You can use the Cluster Admin GUI (see the section “Stopping RMS” on page 133) or you can enter the following command to stop RMS on all nodes from any machine in the cluster:

```
# hvshut -a
```

4.2 Creating a configuration

- ▶ Enter the following command to generate the wizard menu for the configuration example, `mydemo`:

```
# hvw -n mydemo
```

This will create an RMS configuration file named `mydemo.us` in the `/opt/SAW/SAWRrms/build/` directory. If you choose a different name and location, the combined length of the file name and path should not exceed 80 characters.

The *RMS configuration menu* appears, displaying the name of the configuration at the top of the menu (Figure 20).

```
fuji2: Main configuration menu, current configuration: mydemo
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:
```

Figure 20: Main configuration menu

4.3 Adding hosts to the cluster

Before you configure an application, you must define the cluster so that it includes all hosts on which the application may run. The names of all possible RMS hosts should have already been added to the `/etc/hosts` file (see “Appendix—Site preparation” on page 213).

Select the nodes to be included in the configuration. See “Appendix—Cluster planning worksheet” in the *PRIMECLUSTER Installation Guide* for your operating system.

This example assumes `/etc/hosts` contains the following entries, which follow the RMS naming convention:

```
# host names for RMS
192.168.10.83    fuji2RMS
192.168.10.84    fuji3RMS
```

i By default, RMS host names are of the form `<hostname>RMS` to follow the RMS naming convention. To override the default RMS name for a machine, edit that host's `hvenv.local` file and set the `RELIANT_HOSTNAME` variable to the desired name. The contents of that host's `RELIANT_HOSTNAME` variable must match the corresponding `/etc/hosts` entry on every host in the cluster. This must be done before you start the Wizard Tools (`hvw`). If RMS is running, you must also restart RMS.

In this step, you will add the RMS hosts to the cluster.

- At the *Main configuration menu*, enter the number 16. The *Add hosts to a cluster* menu appears (Figure 21).

```
Creation: Add hosts to a cluster:
Current set:
1) HELP
2) QUIT
3) RETURN
4) FREECHOICE
5) ALL-CF-HOSTS
6) fuji2RMS
7) fuji3RMS
Choose the host to add: 7
```

Figure 21: Add hosts to a cluster menu

This menu displays the current set of nodes and lists the machines that can be selected. If you select 5) *ALL-CF-HOSTS*, the RMS Wizards add all nodes in */etc/cip.cf* to this configuration. Otherwise, you can add hosts individually from the displayed list.

- ▶ Select *fuji2RMS* by entering the number 6. Select *fuji3RMS* by entering the number 7 (see Figure 21).

At this screen, you can also choose 4) *FREECHOICE*, which will allow you to enter host names that are not listed in the menu.

- ▶ After all host names have been added, use 3) *RETURN* to return to the *Main configuration menu*.

To remove a node, select 17) *RMS-RemoveMachine* from the *Main configuration menu*. The *Remove hosts from a cluster* menu appears (Figure 22).

```
Removal: Remove hosts from a cluster:
Current set: fuji2RMS fuji3RMS
1) HELP
2) QUIT
3) RETURN
4) ALL
5) fuji2RMS
6) fuji3RMS
Choose the host to remove:
```

Figure 22: Remove hosts from a cluster menu

This menu lists all nodes currently in the cluster. Machines can be removed by selecting them individually or by selecting 4) *ALL* from the menu. In either case, machines being used by one or more applications cannot be removed.

4.4 Creating an application

After you have defined the set of hosts that form the cluster, you can configure an application that will run on those hosts. In this step, we will first create the application using the DEMO turnkey wizard. Begin at the *Main configuration menu* (Figure 23).

```
fuji2: Main configuration menu, current configuration: mydemo
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: 3
```

Figure 23: Main configuration menu

- ▶ **Select *Application-Create* by entering the number 3. The *Application type selection menu* appears (Figure 24).**

```
Creation: Application type selection menu:
 1) HELP
 2) QUIT
 3) RETURN
 4) OPTIONS
 5) DEMO
 6) GENERIC
 7) LIVECACHE
 8) R3ANY
 9) R3CI
10) RTP
Application Type: 5
```

Figure 24: Application type selection menu



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

This example uses the *DEMO* application type, which has been designed to familiarize the user with the configuration process and is intended for demonstration purposes only: other than a few user-specified attributes, everything is preset and ready to run. To configure a real-world application, you would instead select the *GENERIC* application type, as described in the section “Creating a second application” on page 85.

- ▶ Select the *DEMO* application type by entering the number 5.

You have now assigned the *DEMO* application type to your application. This means the *DEMO* turnkey wizard will provide the application with scripts and detectors that were developed for this application type.

There are, however, more parameters to specify before this application can run. One of them might be the application name; you can assign a name of your choice to any application that you configure for RMS. In this case, there is no need to specify an application name, as the *DEMO* wizard provides *APP1* as a default here.

APP1 is a simple application, developed specifically for this example, that generates an animated graphical figure on an X-window display. It will be used demonstrate how an application can be started, stopped, or switched, and how RMS performs failover when the application process is killed on the initial node.

After performing a consistency check, the wizard informs you what to do next (see Figure 25).

```
Consistency check ...
Yet to do: process the basic settings using Machines+Basics
Yet to do: choose a proper application name

Settings of turnkey wizard "DEMO"
1) HELP
2) NO-SAVE+EXIT
3) SAVE+EXIT
4) REMOVE+EXIT
5) ApplicationName=APP1
6) BeingControlled=no
7) Machines+Basics(-)
Choose the setting to process: 7
```

Figure 25: Prompting for further actions

At each step, the wizard checks the consistency of the application being configured. Only consistent applications are allowed to be part of the high-availability configuration.

If you want to specify a different application name, you could do it here by selecting 5) *ApplicationName*. However, because we are using the default of APP1, the *Yet to do* message will disappear after you select 7) *Machine+Basics*.

4.5 Entering Machines+Basics settings

- ▶ Select *Machines+Basics* by entering the number 7. The *Machines+Basics* menu appears (Figure 26).

```

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) (HaltFlag=no)
11) (PreOfflineScript=)                   24) (PartialCluster=0)
12) (OfflineDoneScript=)                  25) (ScriptTimeout=)
13) (FaultScript=)

Choose the setting to process: 5
    
```

Figure 26: Consistency check and Machines+Basics menu

At the top of the menu, the wizard shows you the result of the latest consistency check. The application named *APP1*, which was indicated on the previous screen, has proven to be consistent.

The *Machines[0]* menu item indicates the node where your application will first attempt to come online. In this case, it is *fuji2RMS*.



The RMS Wizards retrieve the default settings for *Machines[0]* from the local node defined in `RELIANT_HOSTNAME`.

Subsequent *Machines[]* items, if any, indicate the list of failover nodes. If the initial node fails, RMS will attempt to switch the application to a failover node, trying each one in the list according to the index order.

At this point, only the initial node appears in the menu, so configure a failover node for your application as follows:

- ▶ Select *AdditionalMachine* by entering the number 5. A menu containing the current list of available nodes appears (Figure 27).

```

1) HELP
2) RETURN
3) fuji2RMS
4) fuji3RMS
Choose a machine for this application: 4

```

Figure 27: List of nodes for failover procedure



The Wizards retrieve the default list of nodes from the CIP configuration file, `/etc/cip.cf`.

Since our application is presently configured for `fuji2RMS`, we will select `fuji3RMS` as the additional node:

- ▶ Select *fuji3RMS* by entering the number 4.

In the menu that follows (Figure 28) you will see your selection confirmed. *fuji3RMS* now appears under *Machines[1]* as the additional node. If there is a failure on `fuji2RMS`, your application is configured to switch over to `fuji3RMS`.

```

Consistency check ...

Machines+Basics (appl:consistent)
 1) HELP                               14) (FaultScript=)
 2) -                                   15) (AutoStartUp=no)
 3) SAVE+EXIT                           16) (AutoSwitchOver=No)
 4) REMOVE+EXIT                          17) (PreserveState=no)
 5) AdditionalMachine                    18) (PersistentFault=0)
 6) AdditionalConsole                    19) (ShutdownPriority=)
 7) Machines[0]=fuji2RMS                 20) (OnlinePriority=)
 8) Machines[1]=fuji3RMS                 21) (StandbyTransitions=)
 9) (PreCheckScript=)                   22) (LicenseToKill=no)
10) (PreOnlineScript=)                   23) (AutoBreak=yes)
11) (PostOnlineScript=)                  24) (HaltFlag=no)
12) (PreOfflineScript=)                  25) (PartialCluster=0)
13) (OfflineDoneScript=)                 26) (ScriptTimeout=)
Choose the setting to process: 16

```

Figure 28: Machines+Basics menu for additional nodes

At this point, the default value of *No* is specified for *16) AutoSwitchOver*. This means that to actually switch your application over, manual action would be required.

To have the switchover procedure carried out automatically, you have to select *16) AutoSwitchOver* in this menu, and then specify the desired mode(s) from the menu that follows (Figure 29).

```

Set flags for AutoSwitchOver:  Currently set: NO (N)
1) HELP
2) -
3) SAVE+RETURN
4) DEFAULT
5) NO(N)
6) HOSTFAILURE(H)
7) RESOURCEFAILURE(R)
8) SHUTDOWN(S)
Choose one of the flags: 6
    
```

Figure 29: AutoSwitchOver mode

- ▶ Set a flag by entering the number 6 for *HOSTFAILURE*. This means that RMS switches an application to another node automatically in the case of a node failure.

```

Set flags for AutoSwitchOver:  Currently set: HOSTFAILURE (H)
1) HELP
2) -
3) SAVE+RETURN
4) DEFAULT
5) NO(N)
6) NOT:HOSTFAILURE(H)
7) RESOURCEFAILURE(R)
8) SHUTDOWN(S)
Choose one of the flags: 7
    
```

Figure 30: Setting flags for AutoSwitchOver mode

- ▶ Enter the number 7 for *RESOURCEFAILURE* (see Figure 30). This means that RMS switches an application to another node automatically in the case of a resource failure.
- ▶ Enter the number 3 for *SAVE+RETURN* (see Figure 30).

You will be returned to the *Machines+Basics* menu (Figure 31). Note that item 16 now displays the *AutoSwitchOver* flags you just set.

```
Consistency check ...

Machines+Basics (appl:consistent)
 1) HELP
 2) -
 3) SAVE+EXIT
 4) REMOVE+EXIT
 5) AdditionalMachine
 6) AdditionalConsole
 7) Machines[0]=fuji2RMS
 8) Machines[1]=fuji3RMS
 9) (PreCheckScript=)
10) (PreOnlineScript=)
11) (PostOnlineScript=)
12) (PreOfflineScript=)
13) (OfflineDoneScript=)
14) (FaultScript=)
15) (AutoStartUp=no)
16) (AutoSwitchOver=HostFailure|ResourceFailure)
17) (PreserveState=no)
18) (PersistentFault=0)
19) (ShutdownPriority=)
20) (OnlinePriority=)
21) (StandbyTransitions=)
22) (LicenseToKill=no)
23) (AutoBreak=yes)
24) (HaltFlag=no)
25) (PartialCluster=0)
26) (ScriptTimeout=)
Choose the setting to process: 3
```

Figure 31: Saving settings

Save your settings now to complete the *Application-Create* process.

- ▶ Select *SAVE+EXIT* by entering the number 3.

4.6 Entering non-basic settings

The DEMO turnkey wizard performs another consistency check before returning you to the wizard settings menu (Figure 32).

```

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

Settings of turnkey wizard "DEMO"
 1) HELP                               11) RemoteFileSystems(-)
 2) -                                   12) IpAddresses(-)
 3) SAVE+EXIT                           13) RawDisks(-)
 4) -                                    14) RC-VolumeManagement(-)
 5) ApplicationName=APP1                15) VERITAS-VoLumeManagement(-)
 6) Machines+Basics(app1)               16) EMC-RdfManagement(-)
 7) CommandLines(-)                     17) FibreCat-MirrorView(-)
 8) Controllers(-)                       18) Gds:Global-Disk-Services(-)
 9) DEMO(-)                              19) Gls:Global-Link-Services(-)
10) LocalFileSystems(-)
Choose the setting to process: 9

```

Figure 32: Non-basic settings



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.

The menu header indicates there is at least one more setting to specify, but it is not a basic setting.

As described earlier, this application creates an animated graphical picture on an X-window display. Therefore, a display setting for the DEMO wizard must be added to the basic settings you have already entered.

- ▶ Select *DEMO* by entering the number 9. The *CommandLines* menu appears (Figure 33).

```
Consistency check ...
Yet to do: set a display

CommandLines (Dem_APP1:not yet consistent)
 1) HELP
 2) -
 3) SAVE+EXIT
 4) REMOVE+EXIT
 5) Display=
 6) StartCommands[0]='hvexec~-F~demo~-c'
 7) StopCommands[0]='hvexec~-F~demo~-u'
 8) CheckCommands[0]=hvdet_demo
 9) (Timeout=300)
10) (AutoRecover=no)
11) (MonitorOnly=no)
Choose the setting to process: 5
```

Figure 33: Prompting for display specification

The menu header indicates that a display still needs to be specified, and the status line tells you that *APP1* is not yet consistent; that is, *APP1* could not yet run with the present *mydemo* configuration.

Because the DEMO wizard has been customized for demonstration purposes, some of the items in the menu have been predefined. Items in the menu body indicate the scripts provided by the wizard for starting, stopping, and checking: see the lines beginning with 6) *StartCommands[0]=*, 7) *StopCommands[0]=*, and 8) *CheckCommands[0]=*.



For technical reasons, spaces are displayed as tildes (~) within the wizard menu commands. The actual commands do not have tildes.

4.7 Specifying a display

Specify the display within the *CommandLines* menu as follows:

- ▶ Select *Display* by entering the number 5. A list of display options appears (Figure 34).

```
1) HELP
2) RETURN
3) FREECHOICE
4) fuji1ADM
5) fuji2ADM
6) fuji3ADM
7) fujiRCA
8) fujiSCON
9) fuji2
10) fuji3
11) fuji2RMS
12) fuji3RMS
Choose a display for this application: 3
      >> 172.25.220.27
```

Figure 34: List of display options

You can choose from the list of detected hosts (all hosts in */etc/hosts*), or you can select 3) *FREECHOICE* to specify an arbitrary host with a suitable display.

- ▶ Select *FREECHOICE* by entering the number 3.

At the *>>* prompt, enter the host name or IP address for the X-window display. In this example, we use the IP address *172.25.220.27*, but you should enter an address in your LAN.

Completing the *FREECHOICE* step initiates another consistency check (Figure 35).

```
Consistency check ...

CommandLines (Dem_APP1:consistent)
  1) HELP
  2) -
  3) SAVE+EXIT
  4) REMOVE+EXIT
  5) Display=172.25.220.27
  6) StartCommands[0]='hvexec~-F~demo~-c~~172.25.220.27'
  7) StopCommands[0]='hvexec~-F~demo~-u~~172.25.220.27'
  8) CheckCommands[0]=hvdet_demo
  9) (Timeout=300)
 10) (AutoRecover=no)
 11) (MonitorOnly=no)
Choose the setting to process: 3
```

Figure 35: Successful consistency check for APP1

The consistency check is successful: you can now use RMS to run *APP1* with the *mydemo* configuration.

Note that the wizard updated the display information for the scripts in items 6) *StartCommands[0]* and 7) *StopCommands[0]*.

This completes the specification of the non-basic settings. You can now save the non-basic settings and exit this part of the configuration procedure.

- From the *CommandLines* menu (Figure 35), select *SAVE+EXIT* by entering the number 3.

This will take you back to the *Settings of turnkey wizard "DEMO"* menu (Figure 36).

```
Consistency check ...

Settings of turnkey wizard "DEMO"
 1) HELP                               11) RemoteFileSystems(-)
 2) -                                   12) IpAddresses(-)
 3) SAVE+EXIT                           13) RawDisks(-)
 4) -                                    14) RC-VolumeManagement(-)
 5) ApplicationName=APP1                15) VERITAS-VolumeManagement(-)
 6) Machines+Basics(app1)              16) EMC-RdfManagement(-)
 7) CommandLines(-)                   17) FibreCat-MirrorView(-)
 8) Controllers(-)                     18) Gds:Global-Disk-Services(-)
 9) DEMO(Dem_APP1)                     19) Gls:Global-Link-Services(-)
10) LocalFileSystems(-)

Choose the setting to process: 3
```

Figure 36: Turnkey wizard DEMO

By specifying the basic and non-basic settings for your application and achieving a consistent result, you have successfully finished the *Application-Create* part of the configuration procedure.

- ▶ Select *SAVE+EXIT* by entering the number 3. This will take you back to the *RMS configuration menu*.

4.8 Activating the configuration

As described in the section “General configuration procedure” on page 49, activating a configuration is the third of the four fundamental steps required to set up a high-availability configuration.

You must stop RMS before activating a configuration. In this example, we stopped RMS before creating the configuration.

The starting point for the activation phase is the *Main configuration menu* (Figure 37).

```
fuji2: Main configuration menu, current configuration: mydemo
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 37: Main configuration menu

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

No further input is required at this stage. As the Wizard completes each task in the activation phase, it displays status information as described in the section “Activating a configuration” on page 59. You will be prompted to continue at the end of the process (see Figure 38).

```
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 38: Successful configuration activation

- ▶ Press the **[Enter]** or **[Return]** key to return to the *Main configuration menu* (Figure 39).

```
fuji2: Main configuration menu, current configuration: mydemo
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 39: Quitting the Main configuration menu

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

This ends the activation phase of the configuration process. At this point, RMS may be started to monitor the newly-configured application.

4.9 Creating a second application

In this section, the *mydemo* configuration is expanded by adding a second application. This example application differs from the first because duplicate configuration procedures are skipped to simplify the example. However, in other parts of the procedure, new features add to the complexity of the *mydemo* configuration.

The second application differs from the first as follows:

- The application uses a new application type, *GENERIC*, instead of *DEMO*. We will use the name *APP2* for the second application.
- *APP2* will control the first application (*APP1*). Therefore, *APP2* must be configured with a *controller* sub-application.

Resume the configuration procedure as follows:

- ▶ Stop RMS if it is running.
- ▶ Return to the *Main configuration menu* with the following command:

```
# hvw -n mydemo
```

The *Main configuration menu* opens (see Figure 40).

```
fuji2: Main configuration menu, current configuration: mydemo
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: 3
```

Figure 40: Starting again with the Main configuration menu

You can add more machines to the cluster at this point, provided the required site preparation steps have been completed.

- ▶ To add machines, select *RMS-CreateMachine* by entering the number 15. Follow the procedure described earlier and then return to the *Main configuration menu* when finished.

From the *Main configuration menu*, select *Application-Create* as follows:

- ▶ Select *Application-Create* by entering the number 3.

The *Application type selection* menu opens (see Figure 41).

```
Creation: Application type selection menu:
 1) HELP
 2) QUIT
 3) RETURN
 4) OPTIONS
 5) DEMO
 6) GENERIC
 7) LIVECACHE
 8) R3ANY
 9) R3CI
10) RTP
Application Type: 6
```

Figure 41: Application type selection menu

This time, assign the *GENERIC* application type to the application. This means that the *GENERIC* turnkey wizard will be in charge of the configuration procedure.

- ▶ Select the *GENERIC* application type by entering the number 6.

After the consistency check, you are prompted to configure the basic settings. *APP2* is the default value for the application name.



If you want to change the name, select 5) *ApplicationName* (see Figure 42).

```

Consistency check ...
Yet to do: process the basic settings using Machines+Basics
Yet to do: choose a proper application name

Settings of turnkey wizard "GENERIC"
1) HELP
2) NO-SAVE+EXIT
3) SAVE+EXIT
4) REMOVE+EXIT
5) ApplicationName=APP2
6) BeingControlled=no
7) Machines+Basics(-)
Choose the setting to process: 7

```

Figure 42: Prompting for further specification

- Select *Machines+Basics* by entering the number 7.

The consistency of APP2 is checked, and the result is positive. When the *Machines+Basics* menu appears, it shows that APP2 is initially configured to run on fuji2RMS (see item 7) *Machines[0]* in Figure 43).

```

Consistency check ...

Machines+Basics (app2:consistent)
1) HELP
2) -
3) SAVE+EXIT
4) REMOVE+EXIT
5) AdditionalMachine
6) AdditionalConsole
7) Machines[0]=fuji2RMS
8) (PreCheckScript=)
9) (PreOnlineScript=)
10) (PostOnlineScript=)
11) (PreOfflineScript=)
12) (OfflineDoneScript=)
13) (FaultScript=)
14) (AutoStartUp=no)
15) (AutoSwitchOver=No)
16) (PreserveState=no)
17) (PersistentFault=0)
18) (ShutdownPriority=)
19) (OnlinePriority=)
20) (StandbyTransitions=)
21) (LicenseToKill=no)
22) (AutoBreak=yes)
23) (HaltFlag=no)
24) (PartialCluster=0)
25) (ScriptTimeout=)
Choose the setting to process: 5

```

Figure 43: Machines+Basics menu

- ▶ Select *AdditionalMachine* by entering the number 5. A menu appears with the list of available machines (Figure 44).

```

1) HELP
2) RETURN
3) fuji2RMS
4) fuji3RMS
Choose a machine for this application: 4
    
```

Figure 44: List of nodes for failover procedure

As with the former application, the additional machine to be specified for the failover procedure is *fuji3RMS*.

- ▶ Select *fuji3RMS* by entering the number 4.

In the screen that follows you see your selection confirmed—menu item 8) *Machines[1]* now displays *fuji3RMS* as the additional machine (Figure 45).

```

Consistency check ...

Machines+Basics (app2:consistent)
 1) HELP                               14) (FaultScript=)
 2) -                                   15) (AutoStartUp=no)
 3) SAVE+EXIT                           16) (AutoSwitchOver=No)
 4) REMOVE+EXIT                         17) (PreserveState=no)
 5) AdditionalMachine                   18) (PersistentFault=0)
 6) AdditionalConsole                   19) (ShutdownPriority=)
 7) Machines[0]=fuji2RMS                20) (OnlinePriority=)
 8) Machines[1]=fuji3RMS                21) (StandbyTransitions=)
 9) (PreCheckScript=)                  22) (LicenseToKill=no)
10) (PreOnlineScript=)                  23) (AutoBreak=yes)
11) (PostOnlineScript=)                 24) (HaltFlag=no)
12) (PreOfflineScript=)                 25) (PartialCluster=0)
13) (OfflineDoneScript=)                26) (ScriptTimeout=)
Choose the setting to process: 3
    
```

Figure 45: Machines+Basics menu

Note that item 16 indicates *AutoSwitchOver=No*, so APP2 will not be switched automatically to *fuji3RMS* if *fuji2RMS* fails: it must be switched manually with the GUI or CLI. To enable automatic switchover, select item 16.

Save your settings and exit this part of the configuration procedure:

- ▶ Select *SAVE+EXIT* by entering the number 3. This takes you to the non-basic settings menu.

4.10 Setting up a controlling application

The basic settings have been specified. However, we still need to set up APP2 to control APP1. This will involve the following two steps, available in the non-basic settings:

- Create a controller object for APP2.
- Specify APP1 as the application to be controlled.

The previous step has taken you to the non-basic settings menu (Figure 46).

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

Settings of turnkey wizard "GENERIC"
1) HELP                                10) RemoteFileSystems(-)
2) -                                    11) IpAddresses(-)
3) SAVE+EXIT                            12) RawDisks(-)
4) -                                    13) RC-VolumeManagement(-)
5) ApplicationName=APP2                 14) VERITAS-VolumeManagement(-)
6) Machines+Basics(app2)               15) EMC-RdfManagement(-)
7) CommandLines(-)                    16) FibreCat-MirrorView(-)
8) Controllers(-)                      17) Gds:Global-Disk-Services(-)
9) LocalFileSystems(-)                 18) Gls:Global-Link-Services(-)

Choose the setting to process: 8
```

Figure 46: Non-basic settings

- ▶ Select *Controllers* by entering the number 8.

This creates a controller object for APP2 and presents a menu that lets you specify the controller settings (Figure 47).

```

Consistency check ...
Yet to do: assign at least one application to control
Yet to do: configure at least one controlled application without the M flag

Settings of application type "Controller" (not yet consistent)
1) HELP
2) -
3) SAVE+EXIT
4) REMOVE+EXIT
5) ControlPolicy=FOLLOW
6) AdditionalAppToControl
7) (InParallel=)
8) (FaultScript=)
Choose the setting to process: 6

```

Figure 47: Assigning a controller

Note that item 5 allows you to change the controller type. This example will use the default follow controller.

4.11 Specifying controlled applications

Once you specify a controller, the wizard needs to know which application to control.

i When an application becomes the child of a controller, the Wizards adjust some of its attributes automatically. In the case of a Follow controller, the child's *Machines[]* entries are overridden by the settings of the parent.

- Select *AdditionalAppToControl* by entering the number 6.

The menu that appears offers you a list from which to choose an application (Figure 48).

```

1) HELP
2) RETURN
3) FREECHOICE
4) app1
Choose an application to control: 4

```

Figure 48: List of applications to be chosen as controlled applications

The controlled application is APP1, while APP2 is the controlling application. Choose the application to be controlled as follows:

- ▶ Select *APP1* by entering the number 4. The controller flags menu appears (Figure 49).

```
Set flags for (sub) application: appl
Currently set: AUTORECOVER,TIMEOUT (AT180)
1) HELP
2) -
3) SAVE+RETURN
4) DEFAULT
5) MONITORONLY(M)
6) NOT:AUTORECOVER(A)
7) TIMEOUT(T)
Choose one of the flags:
```

Figure 49: Menu for setting controller flags

There are a number of flags that can be set for a controlled application. In this example, the *A* (*AUTORECOVER*) flag has been set. The *A* flag means If the controlled application becomes offline, the controlling application tries to restart it. The *AUTORECOVER* menu item is now in the opposite state; that is, ready to be toggled to *NOT*.

The *T* (*TIMEOUT*) flag limits the amount of time tolerated while bringing the controlled application online. In this example, we will reduce the timeout period to 150 seconds.

- ▶ Change the timeout period by entering 7.
- ▶ In the menu that appears (Figure 50), select *FREECHOICE* by entering the number 3.

```
1) HELP
2) RETURN
3) FREECHOICE
4) 180
Set an appropriate timeout: 3
  >> 150
```

Figure 50: Changing controller timeout period

- ▶ At the >> prompt, enter 150 for the timeout period.

- ▶ Press **[Enter]** or **[Return]** to return to the menu for controller flags (Figure 51).

```
Set flags for (sub) application: appl
Currently set: AUTORECOVER,TIMEOUT (AT150)
1) HELP
2) -
3) SAVE+RETURN
4) DEFAULT
5) MONITORONLY(M)
6) NOT:AUTORECOVER(A)
7) TIMEOUT(T)
Choose one of the flags: 3
```

Figure 51: Saving flags for controller

After completing the settings, save them and return to the *Controllers* menu as follows:

- ▶ Select *SAVE+RETURN* by entering the number 3.

The *Controllers* menu shows that the controller settings are now consistent (Figure 52).

```
Consistency check ...

Settings of application type "Controller" (consistent)
1) HELP
2) -
3) SAVE+EXIT
4) REMOVE+EXIT
5) ControlPolicy=FOLLOW
6) AdditionalAppToControl
7) Controllers[0]=AT150:appl
8) (InParallel=)
9) (FaultScript=)
Choose the setting to process: 3
```

Figure 52: Indication of flags set for controller

Note that your settings are confirmed on item 7) *Controllers[0]*: the *A* and *T* flags have been set for APP1.

- ▶ Select *SAVE+EXIT* by entering the number 3.

This takes you back to the *GENERIC* menu (Figure 53).

```

Consistency check ...

Settings of turnkey wizard "GENERIC"
 1) HELP                               10) RemoteFileSystems(-)
 2) -                                   11) IpAddresses(-)
 3) SAVE+EXIT                           12) RawDisks(-)
 4) -                                   13) RC-VolumeManagement(-)
 5) ApplicationName=APP2                14) VERITAS-VolumeManagement(-)
 6) Machines+Basics(app2)              15) EMC-RdfManagement(-)
 7) CommandLines(-)                   16) FibreCat-MirrorView(-)
 8) Controllers(Ct1_APP2)               17) Gds:Global-Disk-Services(-)
 9) LocalFileSystems(-)                 18) Gls:Global-Link-Services(-)

Choose the setting to process: 3

```

Figure 53: Menu with settings for *GENERIC* turnkey wizard

In the *GENERIC* menu, item 8 *Controllers* now displays a controller assigned to APP2.

- ▶ Select *SAVE+EXIT* by entering the number 3. This takes you back to the *Main configuration menu* (Figure 54).

```

fuji2: Main configuration menu, current configuration: mydemo
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:

```

Figure 54: Main configuration menu

This completes the creation of the second application.

4.12 Activating the configuration a second time

After returning to the *Main configuration menu*, you must activate the *mydemo* configuration for the second time. This has to be done because you have modified the configuration by adding another application.

RMS cannot be running while you activate a configuration. In this example, we stopped RMS before creating the second application.

To activate the configuration, begin at the *Main configuration menu* (Figure 55).

```

fuji2: Main configuration menu, current configuration: mydemo
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 55: Main configuration menu

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

No further input is required at this stage. As the Wizard completes each task in the activation phase, it displays status information as described in the section “Activating a configuration” on page 59. You will be prompted to continue at the end of the process (see Figure 48).

```

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 56: Activating the configuration for the second time

- Press the **[Enter]** or **[Return]** key to return to the *Main configuration menu* (Figure 57).

```
fuji2: Main configuration menu, current configuration: mydemo
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 57: Return to Main configuration menu

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

This ends the activation phase of the configuration process.

4.13 Starting RMS

At this point, you are ready to start RMS on all nodes in the cluster to monitor both applications. You can use the Cluster Admin GUI (see the section “Starting RMS” on page 128) or you can enter the following command from any machine in the cluster:

```
# hvcm -a mydemo
```

Note that you do not have to specify “mydemo” on the `hvcm` command line if it is the configuration that was most recently activated.

This ends the configuration example.

5 Administration

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

This chapter contains the following major topics:

- “Overview” on page 97
- “Using Cluster Admin” on page 98
- “Viewing RMS status and attributes” on page 105
- “Controlling RMS operation” on page 128
- “Related administrative procedures” on page 154
- “Using RMS graphs” on page 154

5.1 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.

Most of the GUI examples in this chapter show clusters with typical PRIMECLUSTER product installations. The appearance of some tabs and menus in the GUI display will change according to the products installed for your platform or market.



PRIMECLUSTER Configuration Services (PCS) is not installed when the Wizard Tools are installed.

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 the 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`).

- The CLI procedures presented here are not intended to be a complete description of the commands that are employed. 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—List of manual pages” on page 245.

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.

5.2 Using Cluster Admin

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



Windows desktop systems require the Java plug-in as specified in the *PRIMECLUSTER Installation Guide* for your operating system.

5.2.1 Starting Cluster Admin

- ▶ 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 named FUJI has `fuji2` and `fuji3` as its primary and secondary management servers, the URL would be either one of the following:

- `http://fuji2:8081/Plugin.cgi`
- `http://fuji3:8081/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 the *PRIMECLUSTER Installation Guide* for your operating system.

5.2.2 Logging in

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 the *PRIMECLUSTER Installation Guide* for your operating system.

After the Web-Based Admin View login window appears (Figure 58), log in as follows:

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

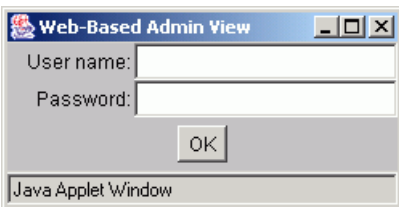


Figure 58: Web-Based Admin View login

After you log in, the Web-Based Admin View window appears (Figure 59).

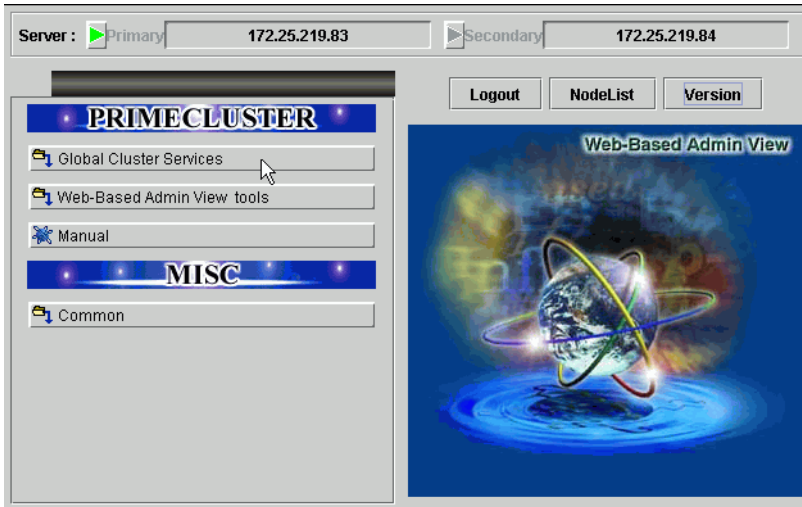


Figure 59: Invoking the Cluster Services GUI

- ▶ Click the *Global Cluster Services* button. The window in Figure 60 appears.

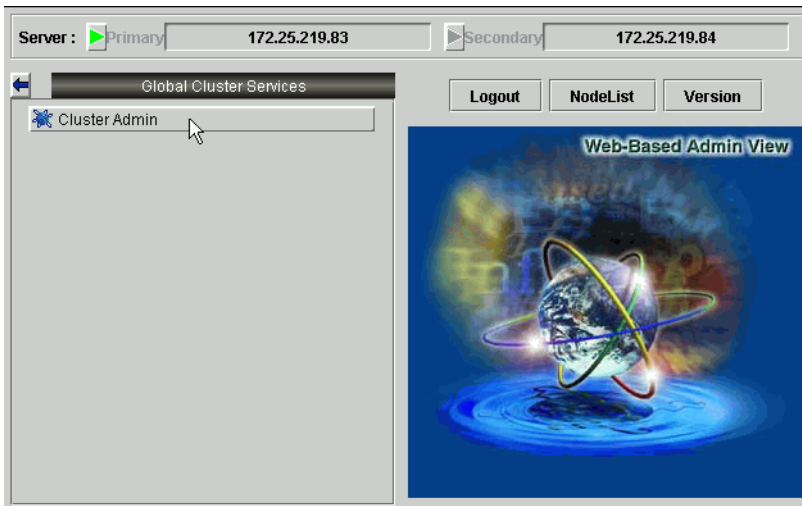


Figure 60: Invoking Cluster Admin

- ▶ Click the *Cluster Admin* button.

The *Choose a node for initial connection* window appears (Figure 61).

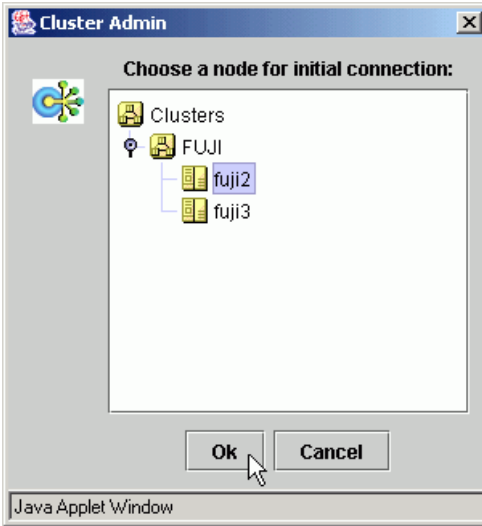


Figure 61: Cluster Admin initial connection menu

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 on *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.



PRIMECLUSTER trusted applets are signed by Verisign, Inc.

The first time you start the Cluster Admin applet, a *Java Security Warning* dialog allows you to choose the security level for the current and future sessions (Figure 62).

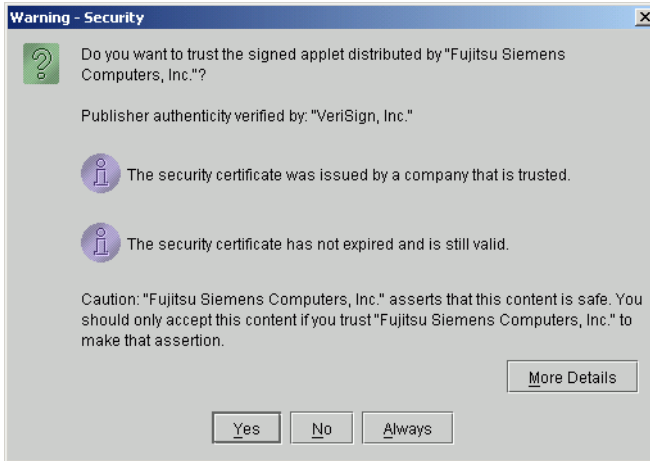


Figure 62: Security certificate dialog

- ▶ Use one of the buttons at the bottom of the dialog to continue your session:
 - *Yes*—The applet operates in trusted mode for the current session only. You will have to respond to the same dialog the next time the applet is started.
 - *No*—The applet operates in untrusted mode for the current session only, so you cannot use the clipboard or other local system resources from the applet window. You will have to respond to the same dialog the next time the applet is started.
 - *Always*—The applet operates in trusted mode for this and all future sessions. The *Java Security Warning* dialog will not appear again.

We recommend that you click on the *Always* button to proceed.

5.2.3 Main Cluster Admin window

When Cluster Admin opens, the initial view is similar to Figure 63.

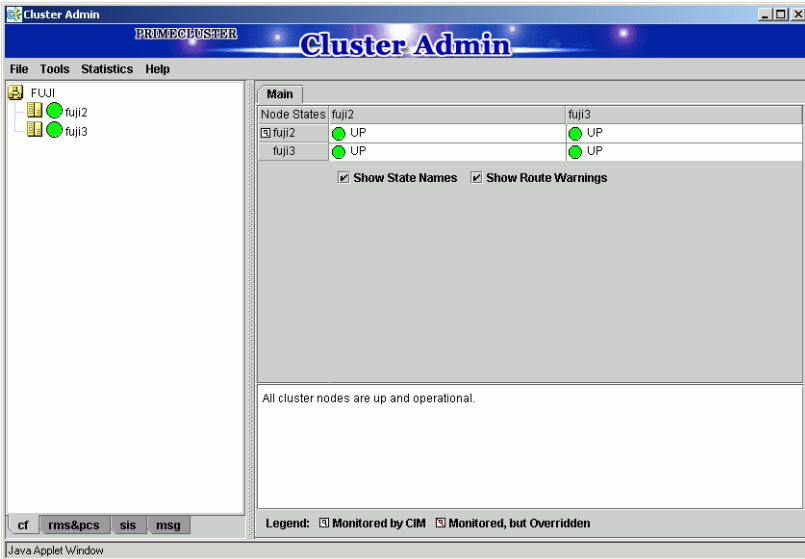


Figure 63: Main Cluster Admin window—Initial view

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

- *cf*
- *rms & pcs* (or simply *rms* if PCS is not installed)
- *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.

5.2.4 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 64).

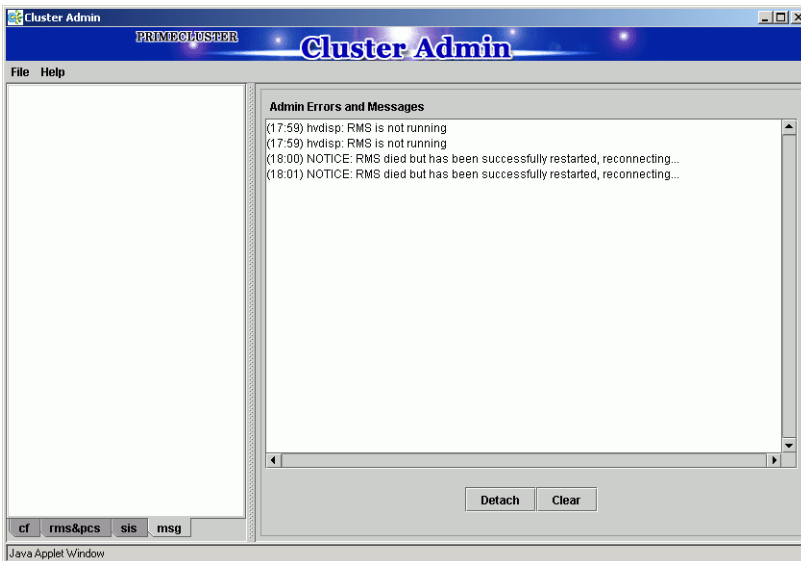


Figure 64: Main Cluster Admin window—message view

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.

5.3 Viewing RMS status and attributes

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&pcs* tab.

A typical RMS view is shown in Figure 65.

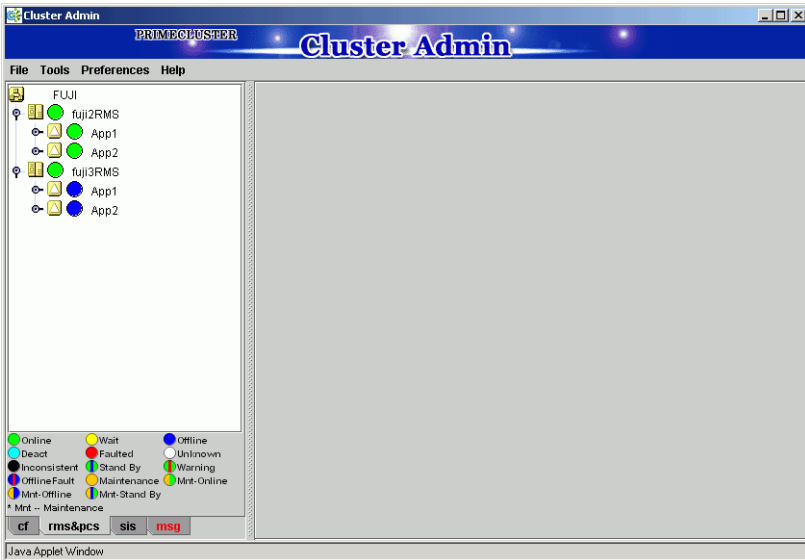


Figure 65: Main Cluster Admin window—RMS view

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.

5.3.1 RMS tree

The 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 66.

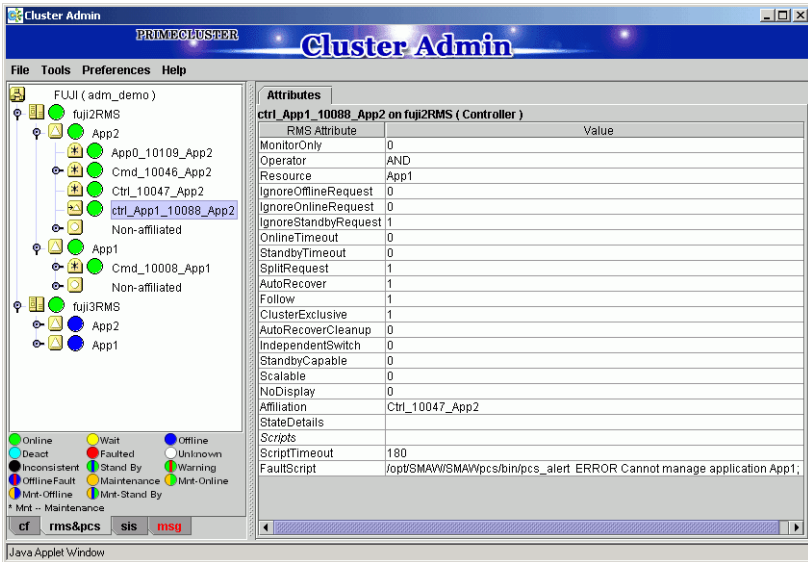


Figure 66: RMS tree with a controller object

In the example shown in the figure, the *Resource* attribute of the highlighted controller object contains the name of the *App1* application: this indicates the dependence of *App2* on *App1*.

A complete visual display of object dependence is available from the RMS graph. See the section “Using RMS graphs” on page 154 for a description of the procedure.

5.3.2 Context-sensitive (pop-up) command menus

You can perform many operations on the RMS tree objects by using the context-sensitive, pop-up command menus. Invoke the pop-up menu by right-clicking on the object. The available menu options are based on the type and the current state of the selected object (Figure 67).

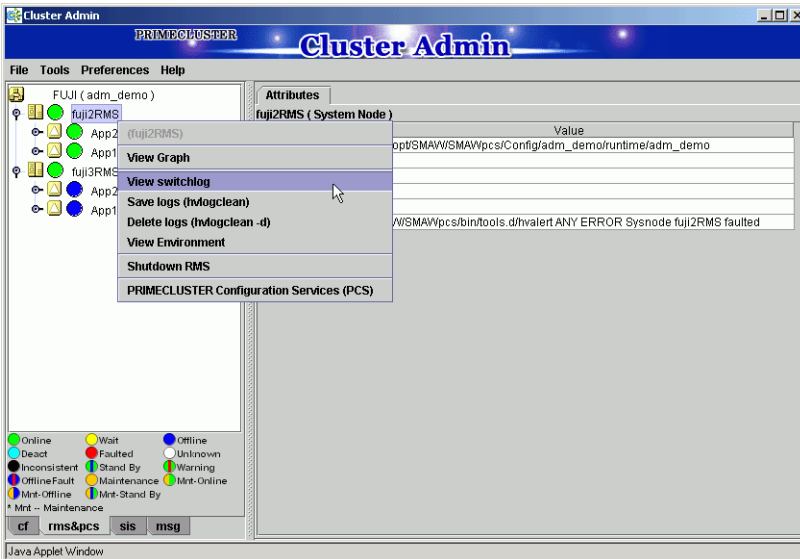


Figure 67: Command pop-up menu for a node

For example, the menu offers different options for a node object selection and an application object selection. It also offers different options for an application object in the online state (Figure 68) and in the offline state (Figure 69).

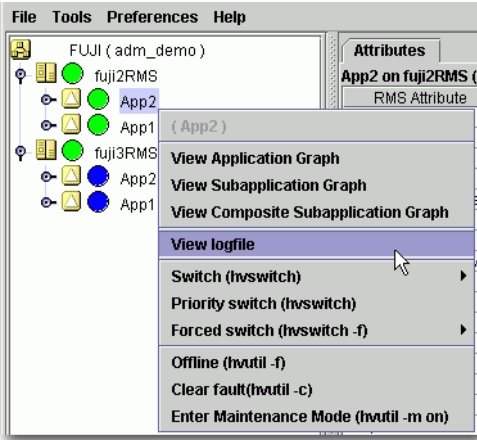


Figure 68: Command pop-up menu for an online application

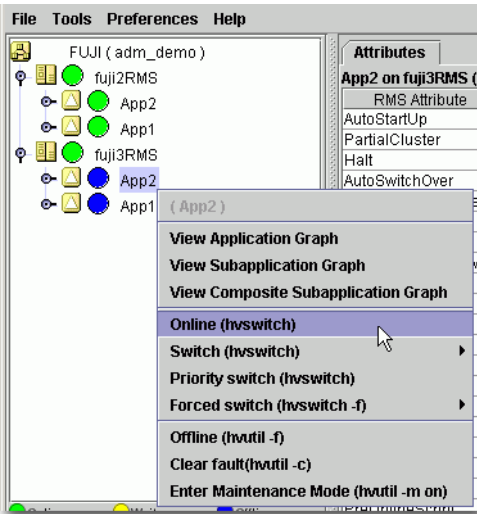


Figure 69: Command pop-up menu for an offline application

5.3.3 Confirmation pop-up windows

When you select an item in an object's context menu that can cause state changes to that object, a confirmation pop-up window appears (Figure 70). To proceed with the action described in the warning message, click *Yes*; to cancel the action, click *No*.

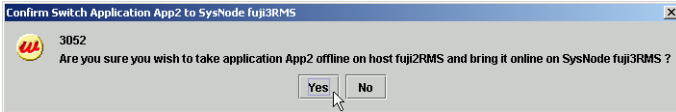


Figure 70: Confirmation pop-up window

For a scalable controlling (parent) application, the confirmation pop-up lists the controlled (child) applications and warns that their states can also change with the specified action (Figure 71).



Figure 71: Confirmation pop-up window for scalable application

5.3.4 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 72).

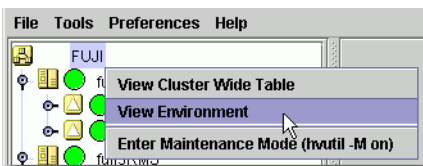


Figure 72: Displaying global environment variables

The global variables will appear under a separate tab in the right pane (Figure 73).

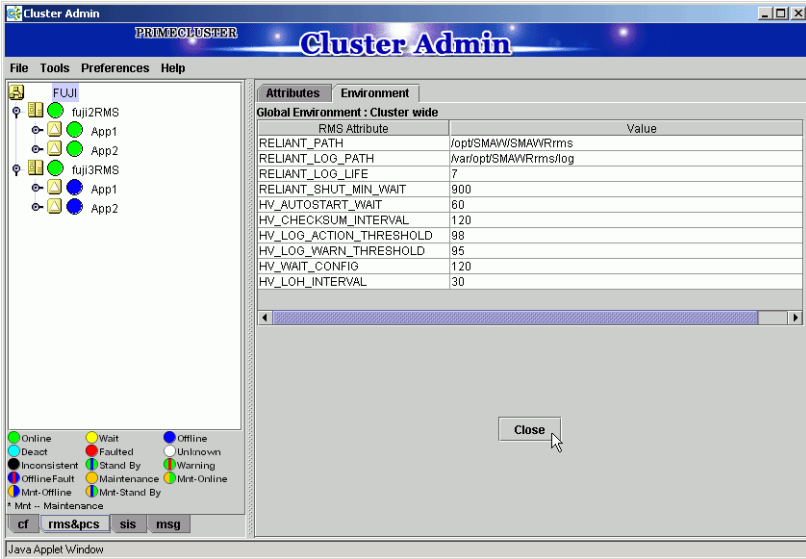


Figure 73: Global environment variable view

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 74).

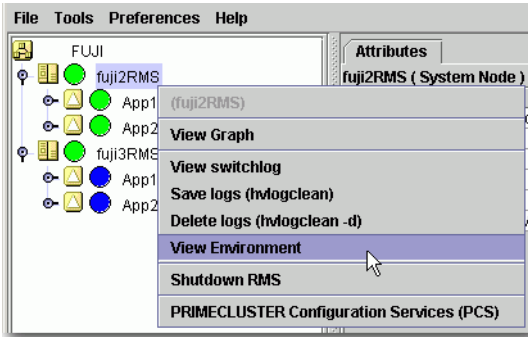


Figure 74: Displaying local environment variables

Both local and global variables will appear under a separate tab in the right pane (Figure 75).

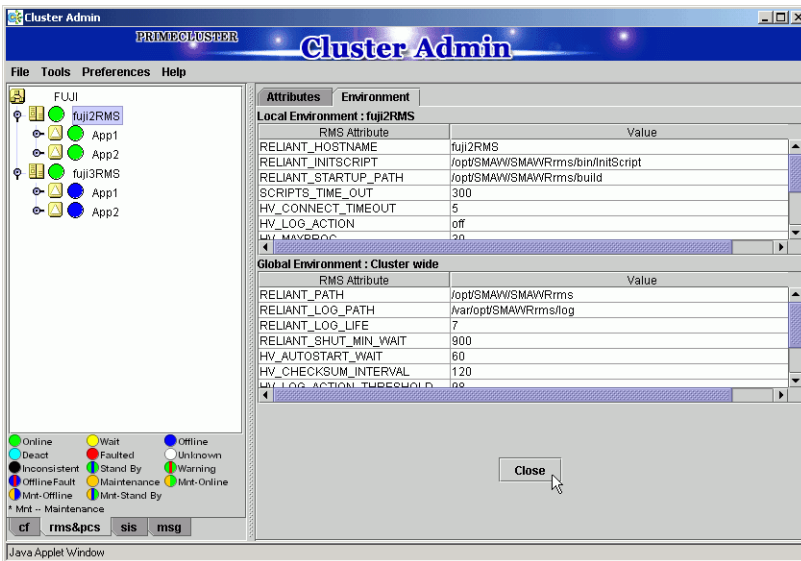


Figure 75: Local environment variables view

CLI: hvdisp

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

```
hvdisp ENV
hvdisp ENVL
```

5.3.5 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 application states appears below the RMS Tree in the left pane of the RMS view (Figure 76).

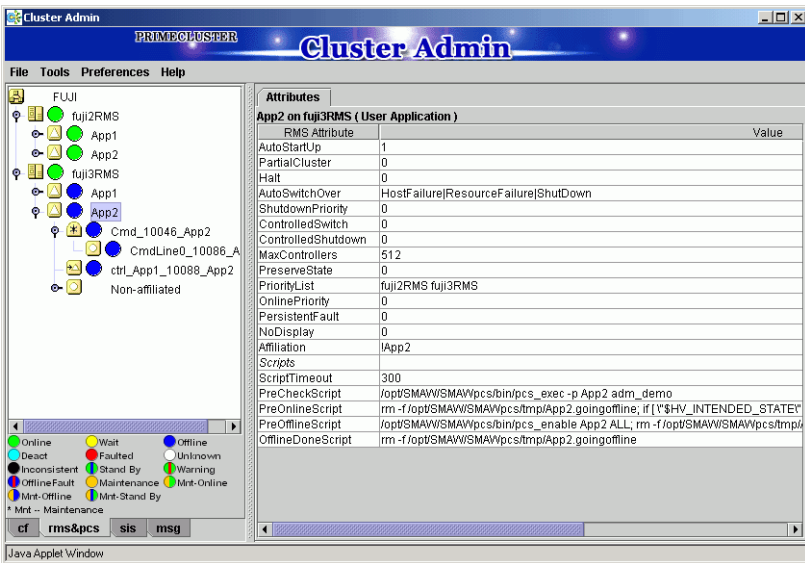


Figure 76: Displaying application and object states

In the example above, the application App2 is online (green status icon) on node fuji2RMS, but offline (blue status icon) on node fuji3RMS.

See “Detectors and states” on page 27 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.

5.3.6 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 77).

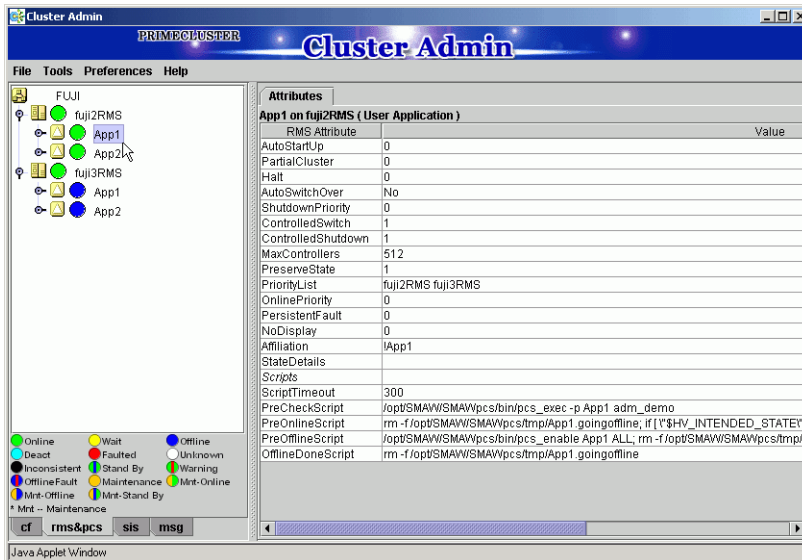



Figure 77: Configuration information or object attributes

5.3.7 Viewing RMS log messages

Using the Cluster Admin interface, you can view two types of RMS log messages on each node: the RMS switchlog, and individual application logs.

 For details about the contents of RMS log files, refer to the *RMS Troubleshooting Guide*.

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 78). Alternatively, select a node and use *Tools -> View switchlog* (Figure 79).

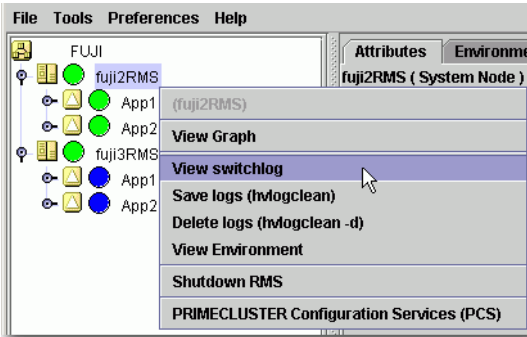


Figure 78: Viewing the RMS switchlog file using a context menu

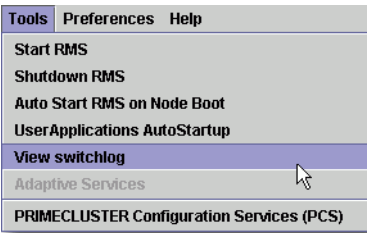


Figure 79: Viewing the RMS switchlog file using the Tools menu

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 80).

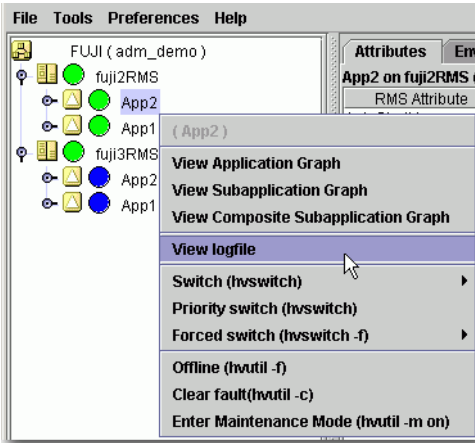


Figure 80: Viewing an application log using a context menu

Each log file is displayed in a separate tab in the right pane (Figure 81).

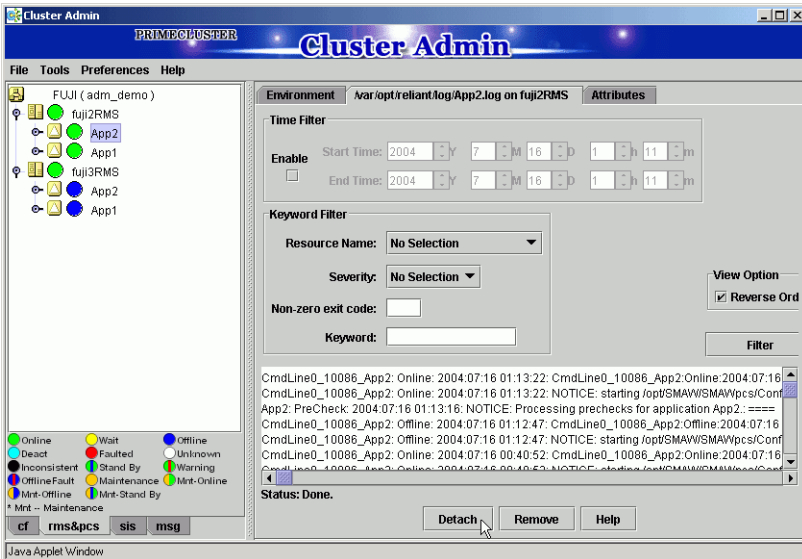


Figure 81: Viewing an RMS log

The *Detach* button at the bottom of the view will separate the current log display tab and display it in its own window (Figure 82). The detached window can be rejoined to the main window with the *Attach* button.

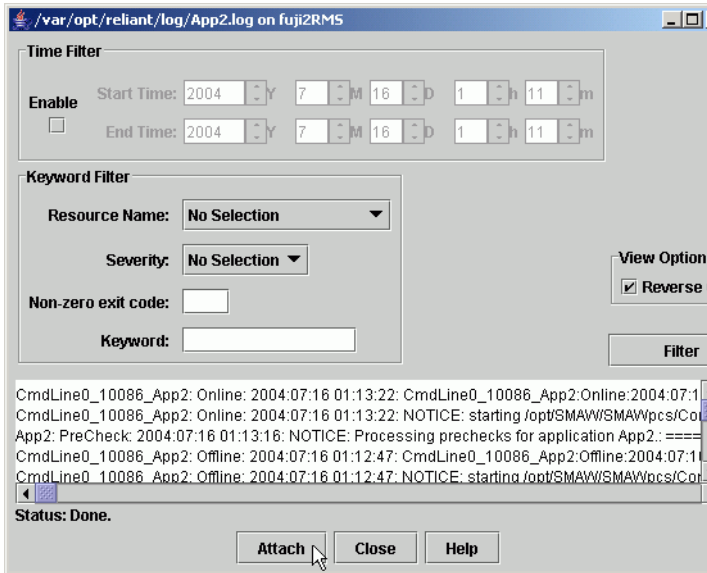


Figure 82: Viewing the RMS switchlog file in a detached window

i All RMS log files, which normally reside in `/var/opt/SMAWRms/Log/`, can be viewed directly using a standard UNIX editor like `vi`.

Common procedures for switchlog and application log viewing

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:

- Timestamp: Click the *Enable* check box and select the period of interest.
- Resource name (for an application only), severity of error messages, non-zero exit code, or keyword. Selected and non-blank criteria are combined with a logical *and*.

i Refer to the *RMS Troubleshooting Guide* for a complete description of severity levels and exit codes.

After you enter your filter criteria, click the *Filter* button to display the filtered log entries.

At any time, you can sort the displayed switchlog entries according to increasing or decreasing time by checking or unchecking the *View in Reverse Order* checkbox in the log viewer window.

Time filter

Figure 83 shows the view for a search based on the date and time.

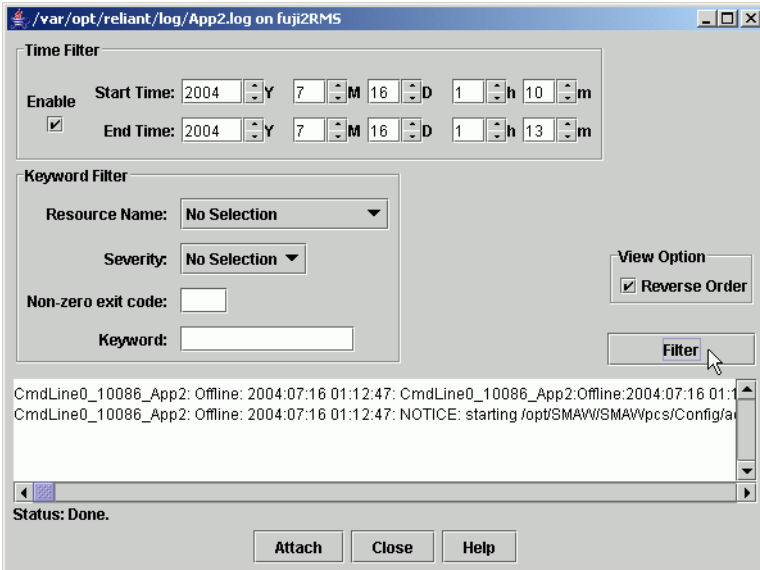


Figure 83: Search based on date and time range

Resource name

Figure 84 shows the view for a search based on a resource name. This search is enabled only in an application log.

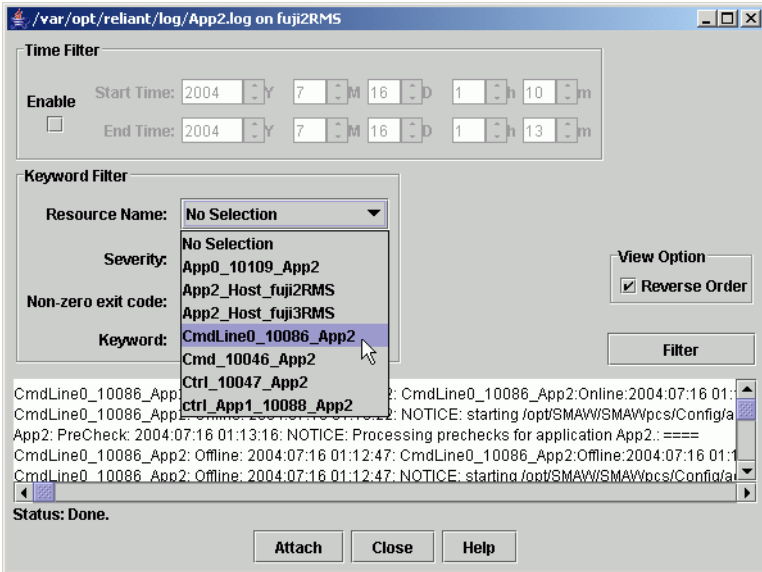


Figure 84: Search based on resource name

Severity

Figure 85 shows the view for a search based on the severity level.

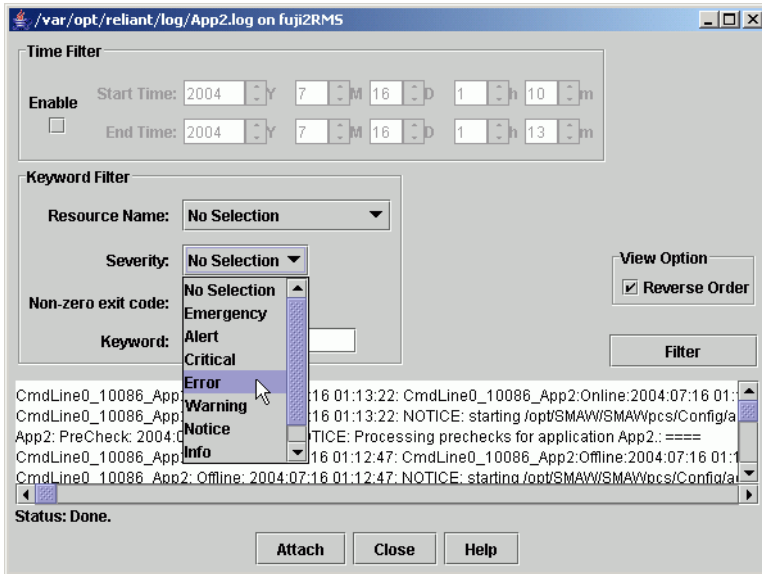


Figure 85: Search based on severity level

Table 5 summarizes the RMS severity levels.

Severity level	Description
<i>Emergency</i>	Systems cannot be used
<i>Alert</i>	Immediate action is necessary
<i>Critical</i>	Critical condition
<i>Error</i>	Error condition
<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

Exit code

Enter a numeric exit code in the *Non zero exit code box*.

Keyword

Enter a string in the *Keyword* box. Special characters and spaces are valid, but wildcards are not interpreted. The search is case-insensitive. Figure 86 shows the view for a search based on a string.

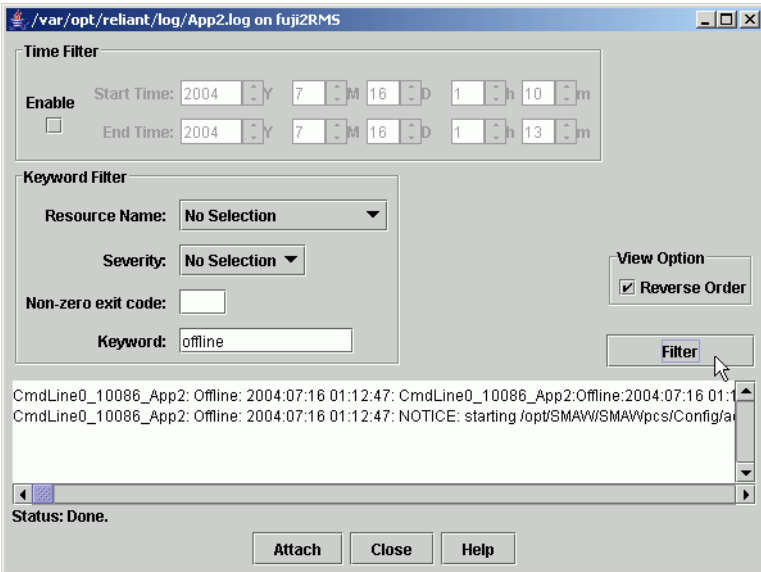


Figure 86: Search based on keyword

You can also search the text in the application log by right-clicking on the displayed text. This brings up a small command pop-up with a *Find* entry that allows you to perform a case-sensitive search (Figure 87).

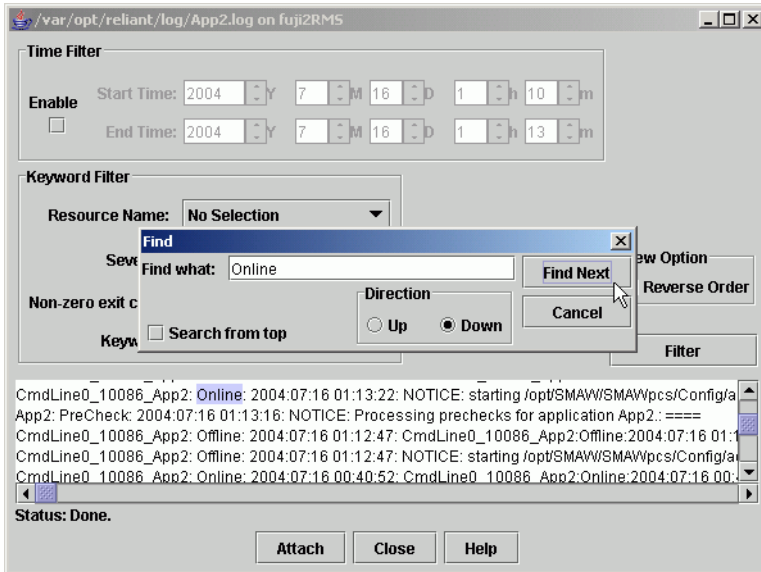


Figure 87: Using the Find pop-up in log viewer

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.

5.3.8 Using the RMS clusterwide table

The RMS clusterwide table displays the state information about application objects in a concise, summary table. The user can see the state of each application on each of the system nodes.

Open the clusterwide table by right-clicking the cluster name (the root of the RMS tree in the left pane) and then selecting *View Cluster Wide table* from the context menu (Figure 88).

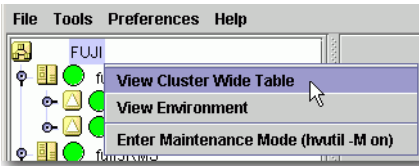


Figure 88: Opening the clusterwide table

The clusterwide table appears in a separate window (Figure 89).

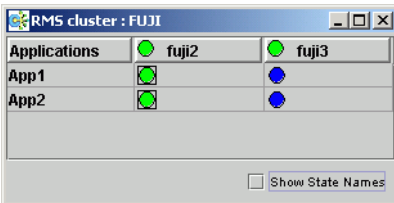


Figure 89: Clusterwide table

Click the *Show State Names* checkbox to display the corresponding state name next to each status icon (Figure 90).

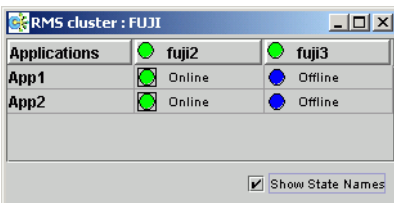


Figure 90: Clusterwide table with state names

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 89 shows that `fujj2` 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 91). This allows the System Administrator to easily spot any `Faulted` applications.

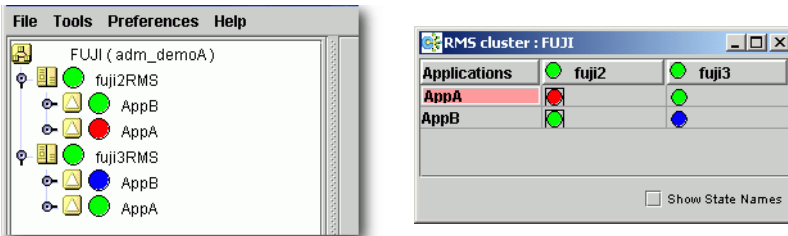


Figure 91: Faulted applications in the clusterwide table

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 92). This alerts the system administrator that some applications are not running anywhere and should probably be brought online on some node.

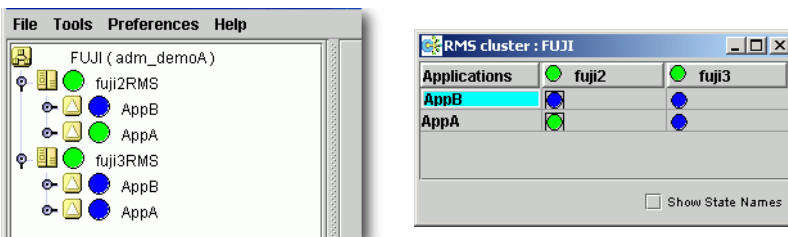


Figure 92: Offline applications in the clusterwide table

If there are both faulted applications and applications that are not online anywhere, then the faulted applications are listed first (Figure 93).

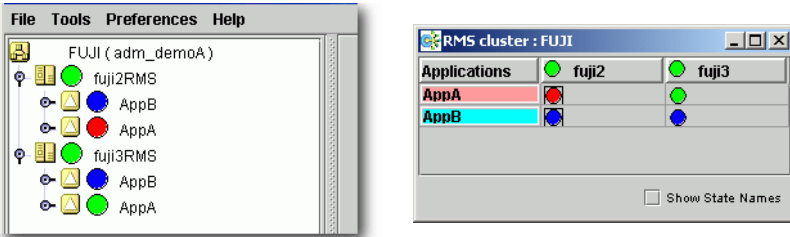


Figure 93: Faulted and offline applications in the clusterwide table

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

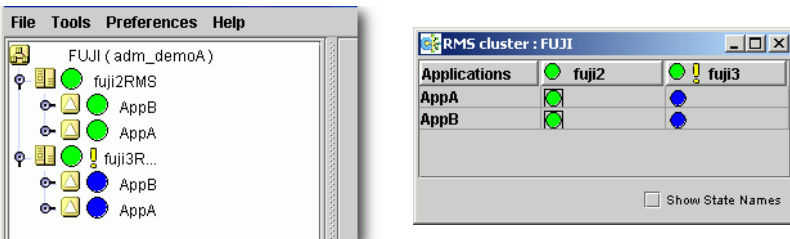


Figure 94: Split-brain conditions in the clusterwide table

5.3.8.1 Using context menus from the clusterwide table

You can use the context-sensitive command pop-up menus to perform some operations on the clusterwide table objects. Invoke the context menu by right-clicking on an object in the table. The menu options are based on the type and the current state of the selected object (Figure 95).

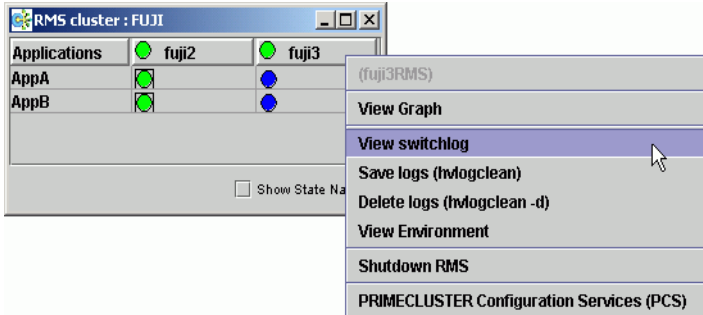


Figure 95: Using command pop-ups in clusterwide table

5.3.9 Display during RMS configuration changes

When you stop and restart RMS with a different configuration, the graphs (described in a later section), the clusterwide table, and the RMS tree are redrawn. In this case, each of the display windows closes and a new display at the same position is displayed.

Figure 96 illustrates the display containing AppA and AppB before RMS is shut down, and Figure 97 shows the RMS GUI after RMS has been restarted with a different configuration that uses app1 and app2.

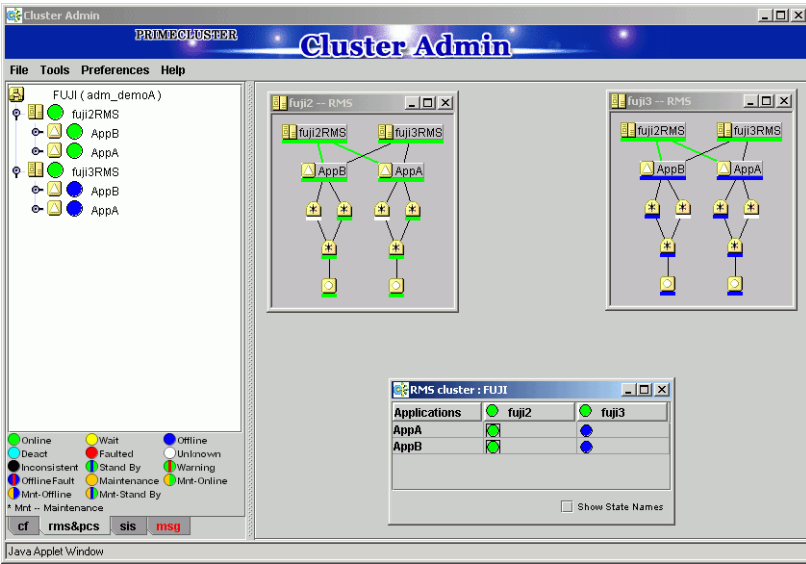


Figure 96: Cluster state before RMS is shut down

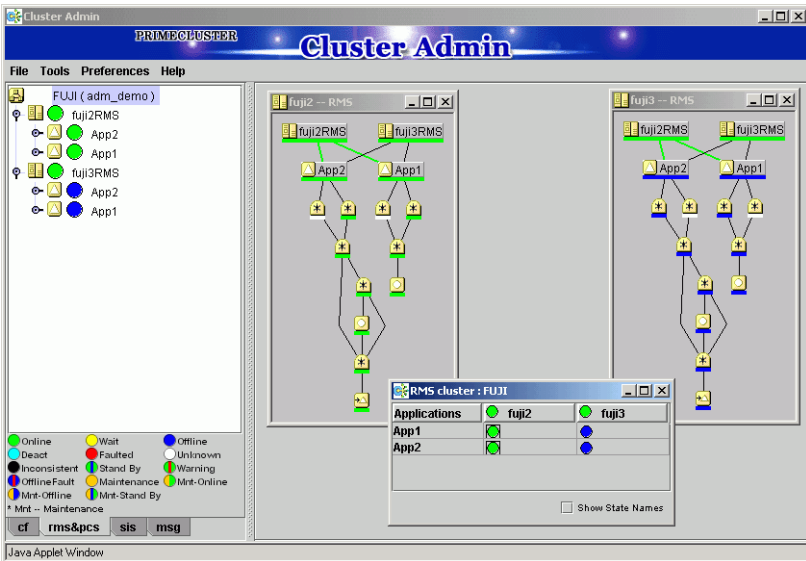


Figure 97: Cluster state after RMS restart with different configuration

5.4 Controlling RMS operation

This section describes basic procedures to control the operation of RMS, including how to start and stop individual nodes and 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 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.

5.4.1 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 subset of nodes that you select.

1. From the Cluster Admin *rms&pcs* (or *rms*) tab, select *Tools* → *Start RMS* (Figure 98).

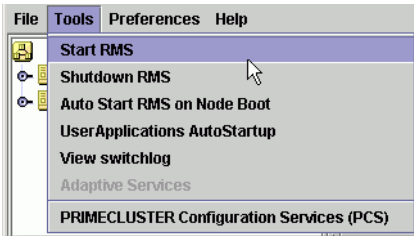


Figure 98: Starting RMS from the main menu

2. The *RMS Start Menu* window opens. To start RMS on all nodes, click the *all available nodes* radio button and then click *OK* (Figure 99).

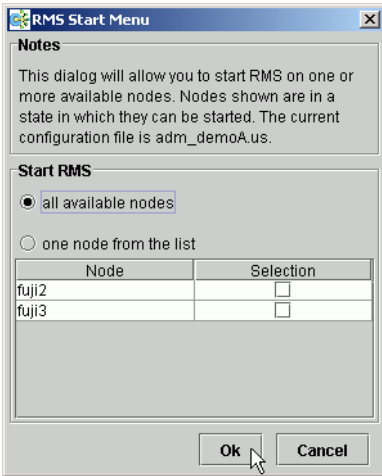


Figure 99: RMS Start Menu for all nodes

3. To start RMS only on selected nodes, click the *one node from the list* radio button. You can select one or more nodes using the checkboxes in the *Selection* column. After making your selections, click *OK* (Figure 100).

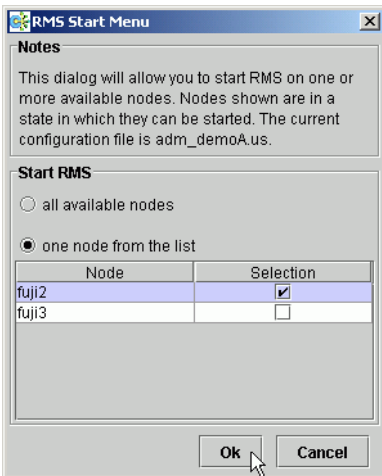


Figure 100: RMS Start Menu for individual nodes

Alternatively, you can start RMS on individual nodes directly from the *Cluster Admin* window:

1. In the left pane, click the *rms&pcs* tab to view the cluster tree.
2. Right-click on the node and select *Start RMS* from the pop-up menu (Figure 101).

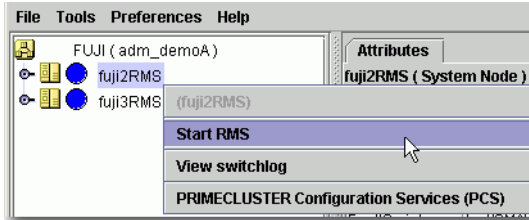


Figure 101: Starting RMS on individual nodes

CLI: hvcm

The syntax for the CLI is as follows:

```
hvcm [-h <recovery_timeout>] [-c config_file] {-a | -s SysNode}
```

The options are:

- h Use the specified heartbeat recovery timeout
- c Use the specified configuration file
- a Start RMS on all nodes in the configuration
- s Start RMS only on the specified node



To change the heartbeat recovery timeout, or start a configuration other than the one most recently activated, you must use the CLI. These parameters cannot be adjusted from the Cluster Admin GUI.

The `hvcm` command 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 default values are sufficient for most configurations.

If `-c` is not present, RMS uses the default startup file `CONFIG.rms`, which starts the most recently activated configuration. This file is located in `<RELIANT_PATH>/etc/`. If the default for the environment variable `RELIANT_PATH` has not been changed, RMS searches for `CONFIG.rms` in `/opt/SMAW/SMAWRrms/etc/`.

The `-h` option sets the UDP heartbeat recovery timeout for the cluster. See “Nodes and heartbeats” on page 11. 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`” on page 234. 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.

5.4.2 Starting RMS automatically at boot time

You can use the following procedure to activate or deactivate automatic RMS startup when the system boots up.

- ▶ From the Cluster Admin *rms&pcs* (or *rms*) tabbed view, select *Tools > Auto Start RMS on Node Boot* (Figure 102).

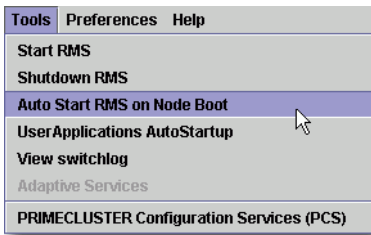


Figure 102: Controlling automatic RMS startup—step 1

You can then choose to activate or deactivate the automatic RMS startup (Figure 103).

i This setting takes effect at the next system startup

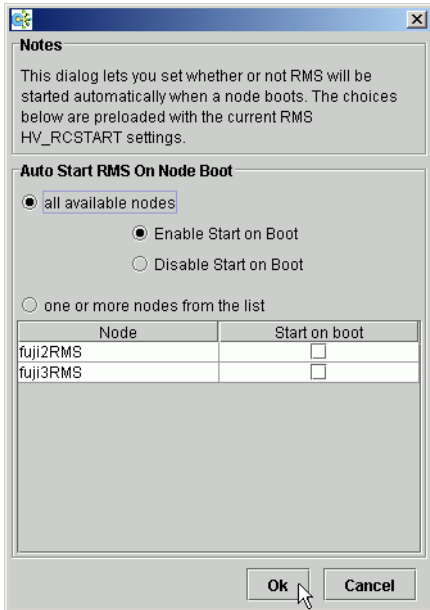


Figure 103: Controlling automatic RMS startup—step 2

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.



The RMS run level is set to the default system run level in `/etc/inittab` when RMS is installed. If the default system run level is later changed, RMS may not start in the correct sequence. You can use the `hvrclv` command to verify or to change the RMS run level. See the `hvrclv` online manual for more information.

5.4.3 Stopping RMS

You can stop RMS on all nodes or on a subset that you select.

Use the *Tools* pull-down menu (Figure 104) and select *Shutdown RMS*.

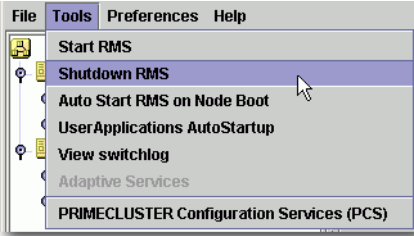


Figure 104: Using the Tools menu to stop RMS

- ▶ To stop RMS on all nodes, click the radio button for *all available nodes* and then click *Ok* (Figure 105).

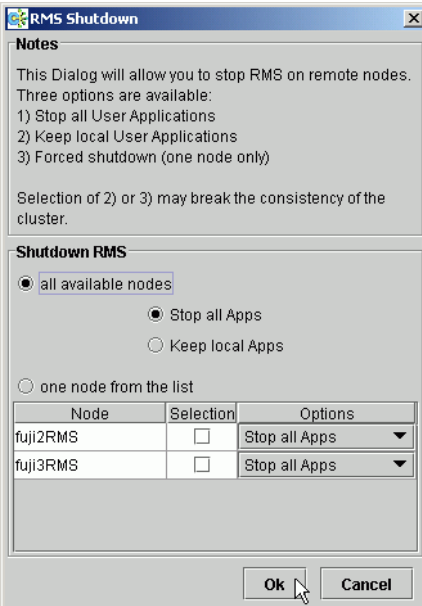


Figure 105: Stopping RMS on all available nodes

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 106).

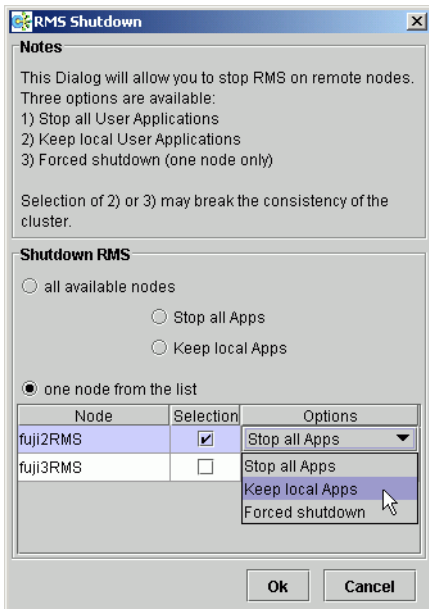


Figure 106: Stopping RMS on one node from the list

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**

Leaving the applications running after stopping RMS or using a forced shutdown can cause data inconsistencies or corruption.

- ▶ Click the *Ok* button to initiate the shutdown with your selections.

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 107).

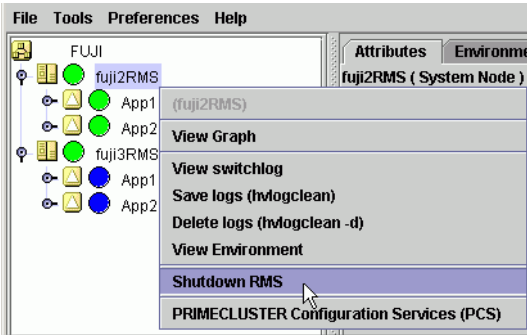


Figure 107: Using the context menu to stop RMS on one node

Only one node will appear in the confirmation window (Figure 108).

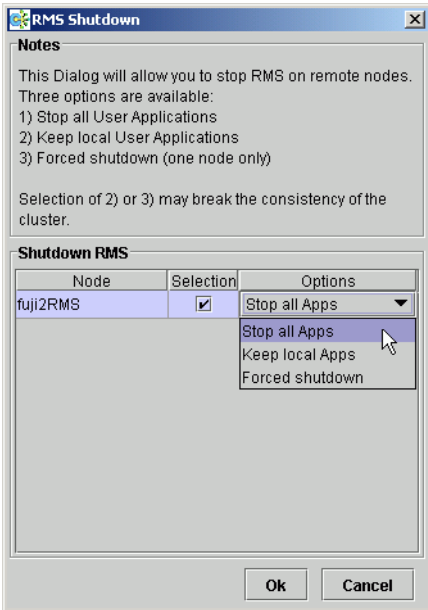


Figure 108: Stopping RMS on one node

Whichever method you use, if you choose to shut down RMS without shutting down the local applications, you will be asked to confirm the operation (Figure 109).

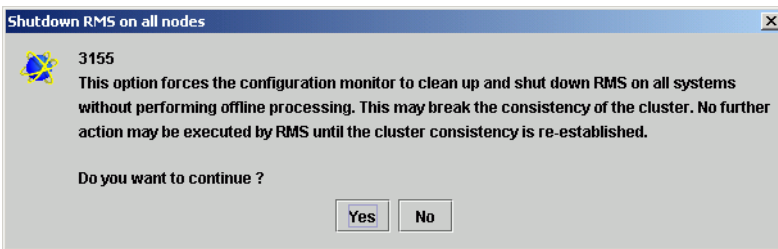


Figure 109: Stopping RMS while keeping applications—confirmation

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 nodes
- A Shut down RMS on all 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 asked to confirm the operation.



Caution

Use the `hvshut -A`, `-f`, and `-L` options carefully as they could result in inconsistencies or data corruption.

5.4.4 Overriding automatic application startup

By default, the automatic startup of each application is controlled by its `AutoStartup` attribute. If `AutoStartup` is set to 1, the application starts automatically when RMS starts or when the application is switched to another node. If `AutoStartup` is set to 0, the application must be started manually.

Automatic application startup can cause problems during some maintenance or troubleshooting procedures. If this is the case, you can suppress the `AutoStartup` attribute for all applications.



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&pcs` (or `rms`) tab, select *Tools* → *UserApplications AutoStartup* (Figure 110).

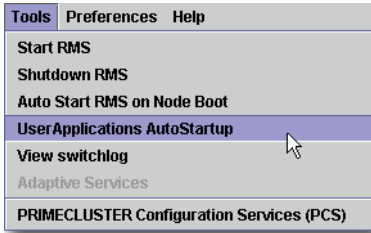


Figure 110: Controlling automatic application startup—step 1

You can then choose to override all `AutoStartUp` settings, or to cancel the override (Figure 111).

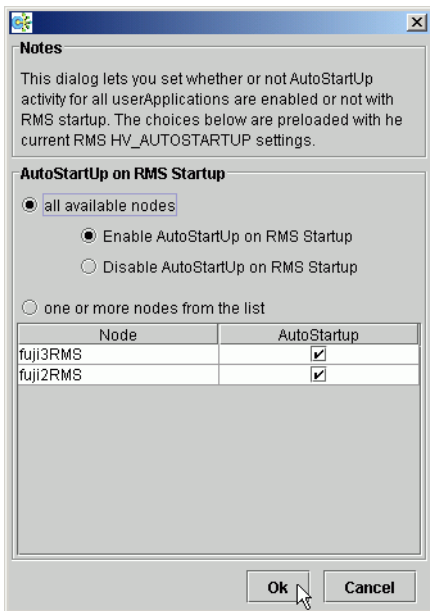


Figure 111: Controlling automatic application startup—step 2

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” on page 238). You can set this variable with the `hvsetenv` command as follows:

```
hvsetenv HV_AUTOSTARTUP [0|1]
```

The allowable values are:

- 0 Prevent automatic application startup at next RMS startup
- 1 Allow automatic application startup at next RMS startup

If no value is specified, the command reports the current value of the HV_AUTOSTARTUP environment variable.

5.4.5 Starting an application

Bring an application online as follows:

Right-click on the application object and select the *Online* option from the pop-up menu (Figure 112).

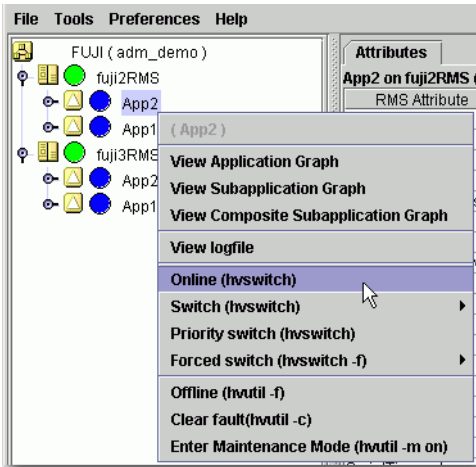


Figure 112: Starting an application

CLI: hvswitch

The syntax for the CLI is as follows:

```
hvswitch [-f] userApplication [SysNode]
```

The `hvswitch` command manually switches control of a `userApplication` resource from one system node to another in the RMS configuration. The resource being switched must be of type `userApplication`. The system node must be of type `SysNode`. If no `SysNode` is specified, the application is switched to the local node. The `-f` option is a forced-switch option.



Caution

Use the `hvswitch -f` option carefully as it could result in inconsistencies or data corruption.

5.4.6 Switching an application

Switch an online application as follows:

- ▶ Right-click on the application object 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 114).

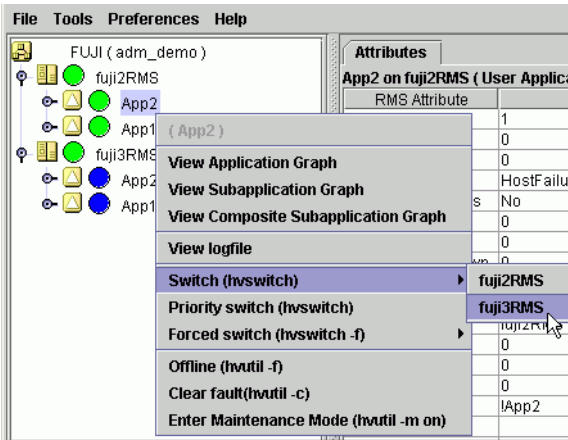


Figure 113: Switching an application

It is recommended that you use the normal mode of switching applications to ensure that application and data consistencies and integrity are maintained.



Caution

Use the forced switch mode only if an application cannot be switched normally. A forced application switch overrides all safety checks and could even result in data corruption or other inconsistencies.

If the application is busy, the command pop-up will not offer the choices to switch the application. Instead, the command pop-up indicates that the application is busy and that you should try later (Figure 115).

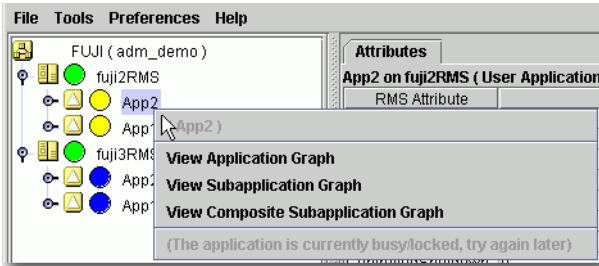


Figure 114: Switching a busy application

CLI: hvswitch

Refer to the section “Starting an application” on page 140 for information on this command.

5.4.7 Taking an application offline

Shut down an online application as follows:

Right-click on the application object and select the *Offline* option from the pop-up menu (Figure 116).

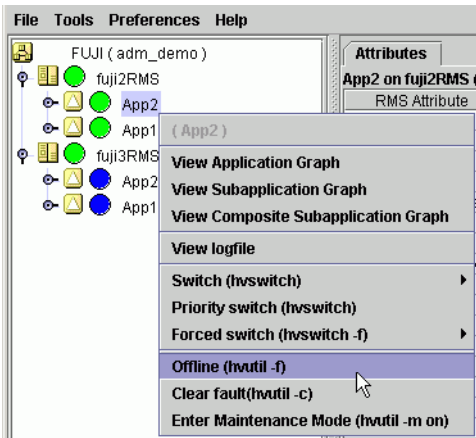


Figure 115: Shutting down an application

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.

5.4.8 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* has nothing to do with activating an RMS configuration—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`’.

5.4.9 Clearing a fault

Clear the fault for an application in the `Faulted` state as follows:

Right-click on the application object and select the *Clear Fault* pop-up menu option (Figure 117).

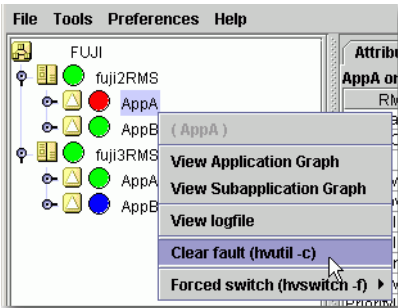


Figure 116: Clearing an application fault

You will be informed of the action to be taken and asked to confirm the operation before the command proceeds (Figure 117).

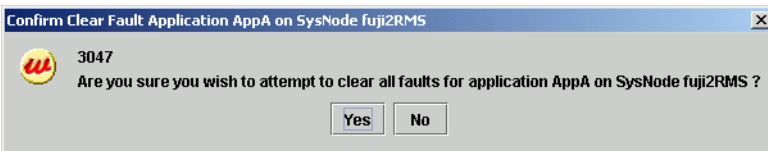


Figure 117: Clearing an application fault—confirmation dialog

See also “Clearing faults in maintenance mode” on page 150.

CLI: `hvutil -c`

The syntax for the CLI is as follows:

```
hvutil -c userApplication
```



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 asked to confirm the operation before the command proceeds.

5.4.10 Clearing a SysNode Wait state

Clear any node in the `Wait` state as follows:

- ▶ Right-click on the node and select the *Online* or *Offline* option from the pop-up menu.

The clearing of the `Wait` state for a node will be ignored unless the Shutdown Facility (SF) timeout has been exceeded.

CLI: `hvutil -o`

The syntax for the CLI is as follows:

```
hvutil -o SysNode
```

This command clears the `Wait` state for the specified `SysNode` on all cluster nodes after the SF failed to kill the cluster node (`SysNode`) by returning the specified `SysNode` to the online state. If the `SysNode` is currently in the `Wait` state, and if the last detector report for the `SysNode` is in the online state, the `Wait` state is cleared and the `SysNode` goes back to the online state as if no kill request had ever been sent.



Caution

Use care when clearing a `SysNode Wait` state manually. Clearing a `Wait` state with `'hvutil -o SysNode'`, `'cftool -k'`, or the GUI causes RMS, CF, and SF to believe that the node in question has been confirmed to be down. Doing so without the node really being down can lead to data corruption.

5.4.11 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.

Entering maintenance mode

You can enter maintenance mode for all applications on all 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 118).

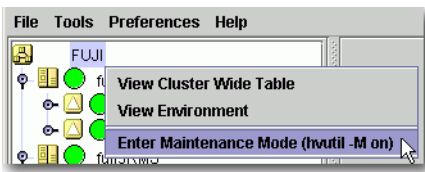


Figure 118: Starting maintenance mode for all applications

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 119).

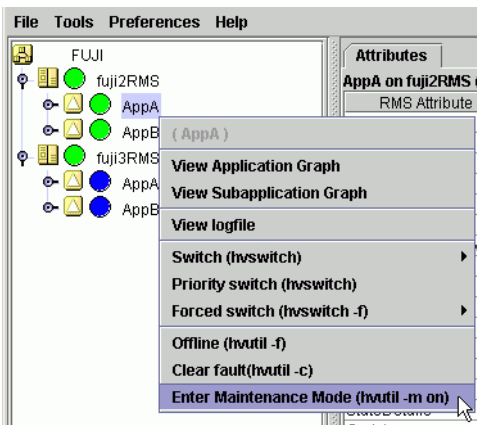


Figure 119: Starting maintenance mode for a single application

In either case, you will be prompted to confirm the operation (Figure 120 and Figure 121).

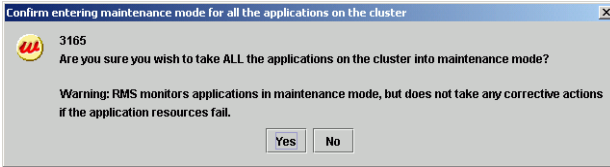


Figure 120: Maintenance mode confirmation for all applications

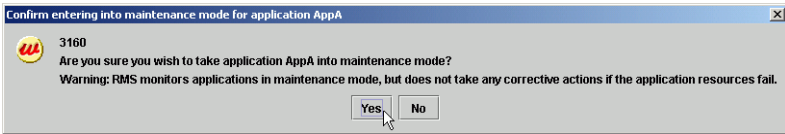


Figure 121: Maintenance mode confirmation for one application


 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.

Figure 122 shows the Cluster Admin window after one application is put into maintenance mode.

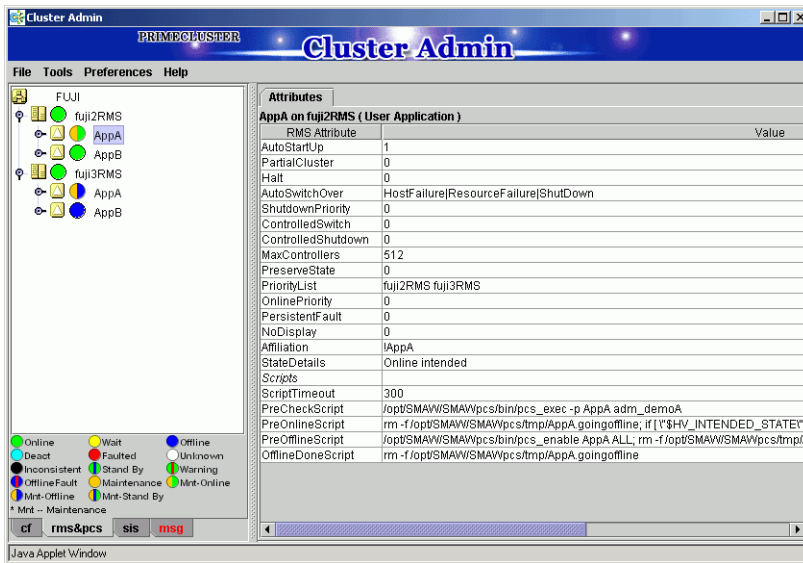


Figure 122: Typical cluster in maintenance mode

Note how the application status icons indicate the intended state (the state that would be attained if the application were taken out of maintenance mode).

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 is the parent and which 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 above example, AppA and AppB are independent. While AppA is in MM, AppB continues to operate normally and can be switched from one node to the other. Compare the application states in Figure 122 above to those in Figure 123 below.

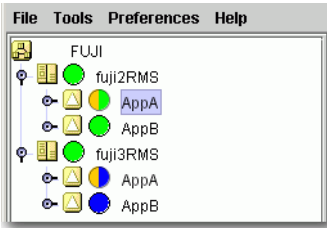


Figure 123: Normal operation of independent application

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 for an application before you can switch that application offline or to another node.
- You must exit MM everywhere in the cluster before you can shut down RMS.

Exiting maintenance mode

To exit maintenance mode, return to the same object you used to enter maintenance mode:

- ▶ Right-click on the cluster or the application and select *Exit Maintenance Mode* from the popup menu (Figure 124 and Figure 125).

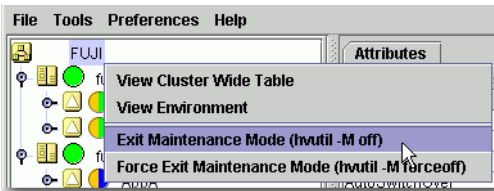


Figure 124: Normal maintenance mode exit for all applications

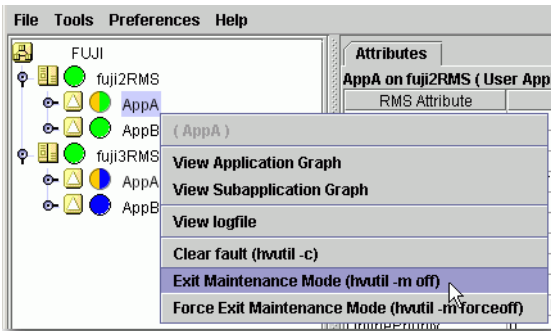


Figure 125: Normal maintenance mode exit for a single application

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.

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 126; the prompt for all applications is similar.

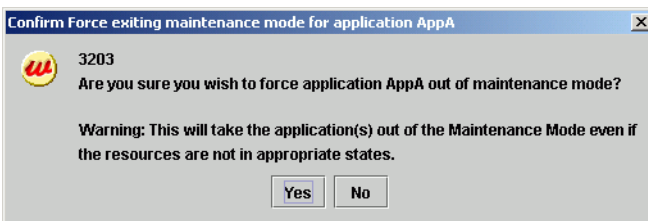


Figure 126: Forced maintenance mode confirmation for all applications

Clearing faults in maintenance mode

If a fault occurs in an application while it is in maintenance mode, you must clear the fault before that application can return to normal operation. (It does not prevent other applications from returning to normal mode.)

RMS marks this type of fault condition with a blue exclamation mark next to the *Maintenance-Online* status icon, and with the state name *Online!!* in the clusterwide table (Figure 127).

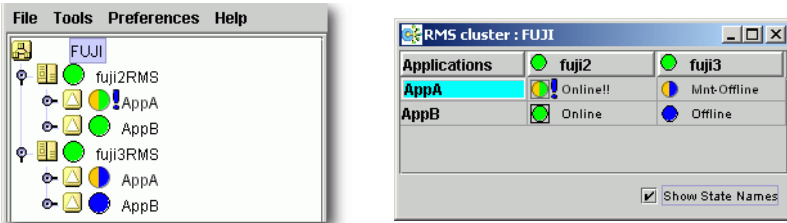


Figure 127: Application with fault condition during maintenance mode

To clear the fault, right-click on the faulted object and choose *Clear fault* in either the configuration tree (Figure 128) or the clusterwide table (Figure 129).

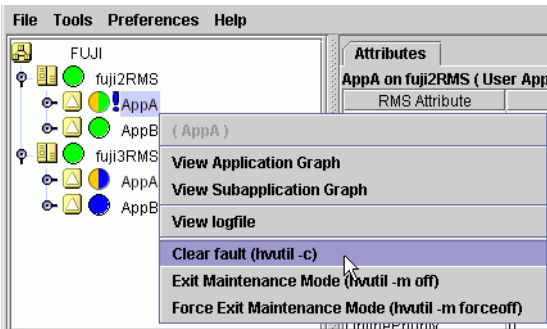


Figure 128: Fault clearing from configuration tree

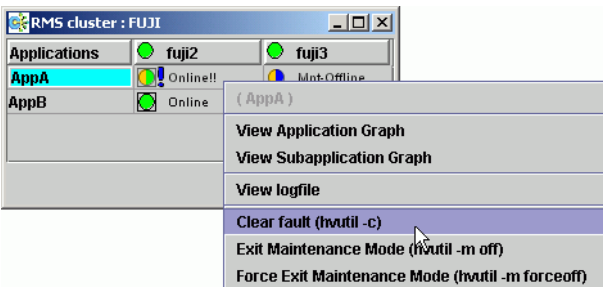


Figure 129: Fault clearing from clusterwide table

After the fault is cleared successfully, the application returns to normal maintenance mode (Figure 130).

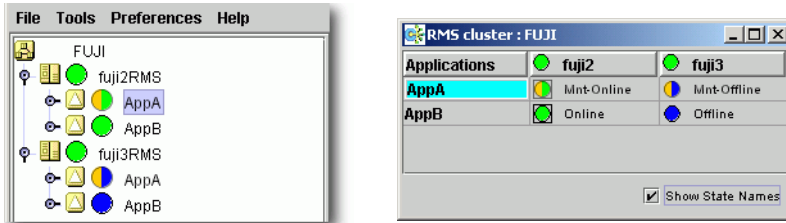


Figure 130: Application returned to normal maintenance mode

In some cases, it may be necessary to select *Force Exit Maintenance Mode* from the context menu. If this is successful, the application will return to normal operation mode with the fault cleared.

CLI: hvutil -m and -M

Control maintenance mode with the `hvutil` command:

```
hvutil -M { on | off | forceoff }
```

```
hvutil -m { on | off | forceoff } userApplication
```

Options:

- `-M` Applies the maintenance mode operation to all applications on all nodes
- `-m` Applies the maintenance mode operation to the specified application on all nodes

Operations:

- `on` Starts maintenance mode
- `off` Stops maintenance mode if all resources are in the appropriate state
- `forceoff` Forces maintenance mode 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.

5.5 Related administrative procedures

The Cluster Admin *Tools* menu provides the following additional entries related to RMS operations:

- *Adaptive Services*

This appears only on Linux platforms. It is disabled (grayed out) if the cluster is not running Adaptive Services.

5.6 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 the graph after RMS is shut down

5.6.1 RMS full graph

You can see the RMS full graph by right-clicking on any system node and selecting *View Graph* from the context menu (Figure 131).

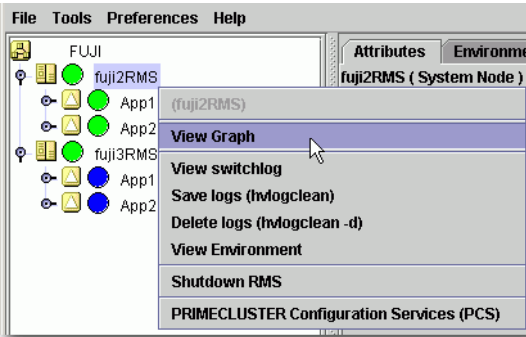


Figure 131: Viewing the RMS full graph on a node

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

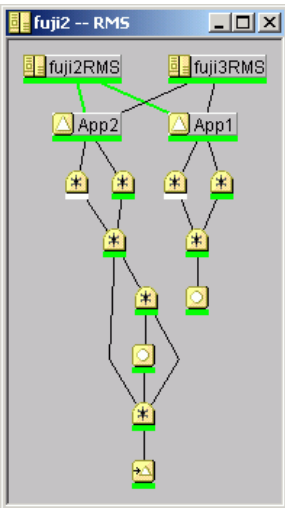


Figure 132: Typical RMS full graph

The RMS full graph (Figure 132) displays the complete RMS configuration of the cluster and represents the following items:

- Relationships between objects
- Dependencies of objects
- Object types, indicated by the object's icon

- Current object state, indicated by the colored bar beneath each icon

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.

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 tooltip (Figure 133).

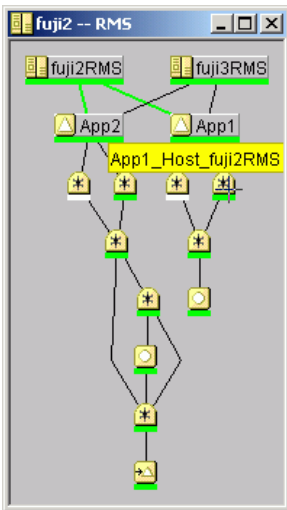


Figure 133: RMS full graph—object tooltip

Clicking on the object brings up a window with further details such as the object's attributes (Figure 134).

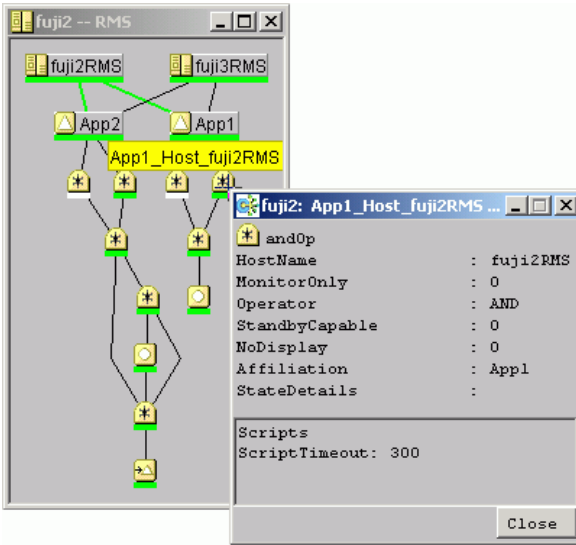


Figure 134: RMS full graph—object details

5.6.2 Application graph

You can see a graph for a single application by right-clicking on an application and selecting *View Application Graph* from the context menu (Figure 135).

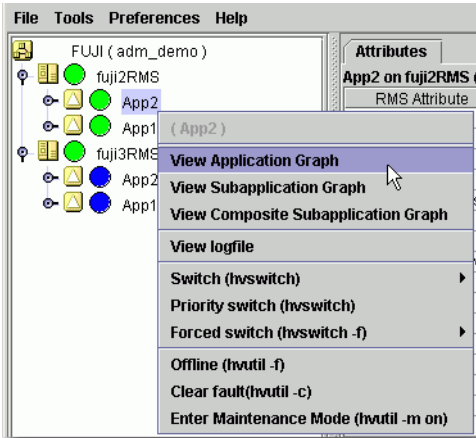


Figure 135: Viewing an RMS application graph

The application graph is similar to the full graph, except that it shows only the selected application and its resources (Figure 136). 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.

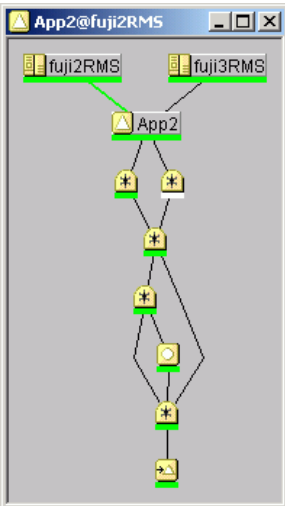


Figure 136: Typical RMS application graph

5.6.3 Subapplication graph

You can see a subapplication graph by right-clicking on an application and selecting *View Subapplication Graph* from the context menu (Figure 137).

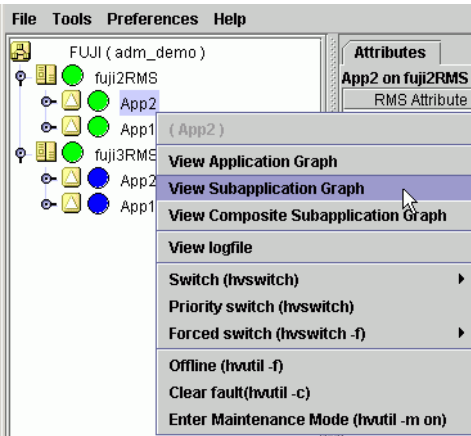


Figure 137: Viewing an RMS subapplication graph

This graph displays all the subapplications used by the selected application, showing the connections between the subapplications (Figure 138).

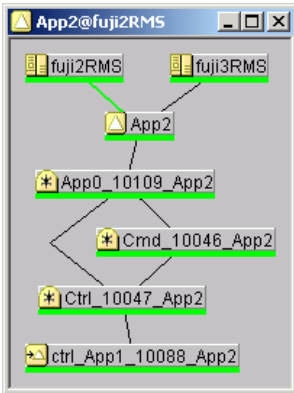


Figure 138: Typical RMS subapplication graph

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

5.6.4 Composite subapplication graph

You can see a composite subapplication graph by right-clicking on an application and selecting *View Composite Subapplication Graph* from the context menu (Figure 139).

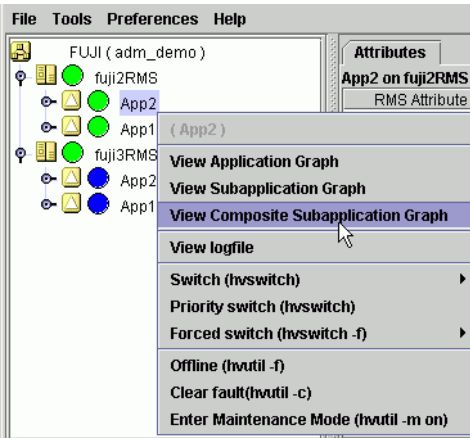


Figure 139: Viewing an RMS composite subapplication graph

The composite subapplication graph is a variation of the subapplication graph for controlled application scenarios: 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 how *App1* appears in the composite subapplication graph in Figure 140 below, and compare this to the standard subapplication graph shown in Figure 138 above.

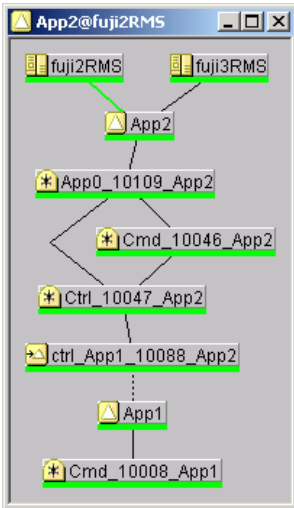


Figure 140: Typical composite subapplication graph

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 application depends on, whether directly or indirectly.

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

5.6.5 Using command pop-up menus from the graph

You can use the context sensitive command pop-up menus on the RMS graph nodes to perform the same operations that are available in the Cluster Admin RMS view. Invoke the pop-up menu by right-clicking on an object. The menu options are based on the type and the current state of the selected object (Figure 141).

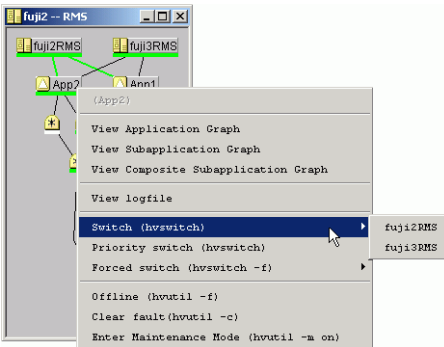


Figure 141: Using a command pop-up menu from the graph

5.6.6 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 resource names, affiliation names, or both to the graphs, use the checkboxes on the *Preferences* menu. Figure 142 shows the preference setting and a corresponding graph that displays affiliation names.

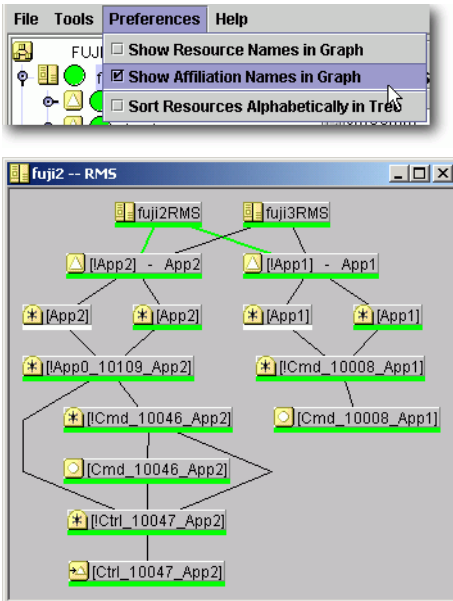


Figure 142: Displaying an RMS graph with affiliation names

Figure 143 shows the preference setting and a corresponding graph that displays resource names.

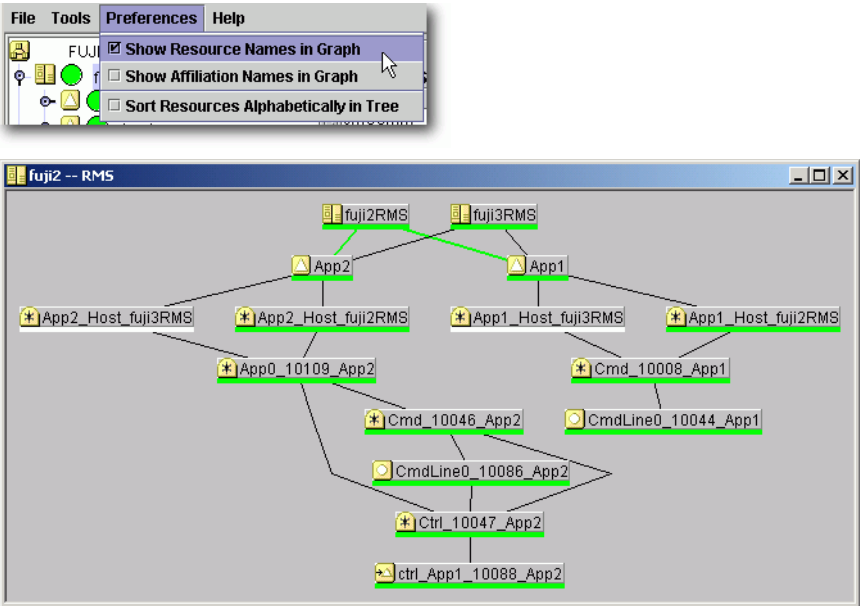


Figure 143: RMS graph with resource names

If both options are selected, graphs will display both the affiliation names and resource names (Figure 144). This combination stretches the graph horizontally, which may make it difficult to read.

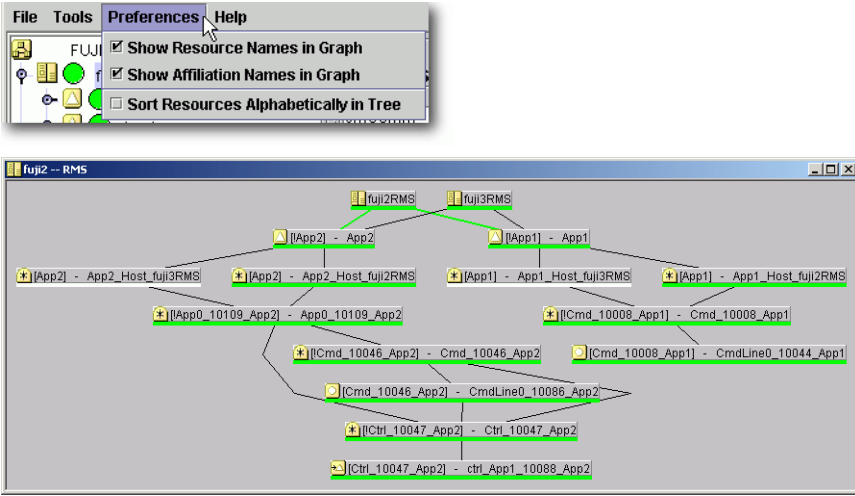


Figure 144: RMS graph with resource and affiliation names

5.6.7 Interpreting the graph 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. The main view and the clusterwide table will continue to show the application states until RMS is shut down on all nodes.

For example, suppose RMS is shut down only on one node (*fujij2*) of our example cluster. Figure 145 shows the full RMS graph obtained from that node.

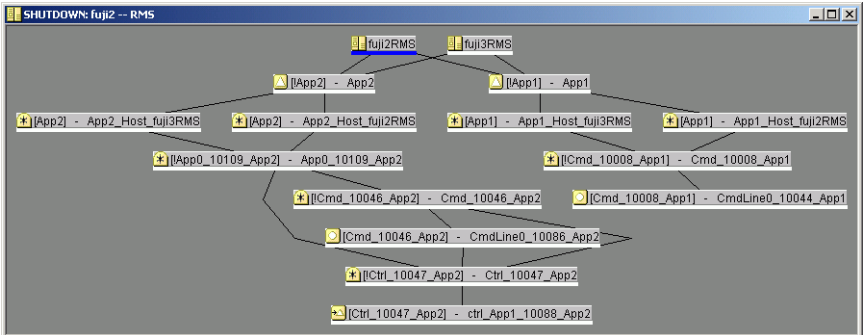


Figure 145: RMS graph after shutdown on one node

However, as long as RMS continues to run on the remaining node, *fujij3*, the RMS main view and the clusterwide table will appear as shown in Figure 146.

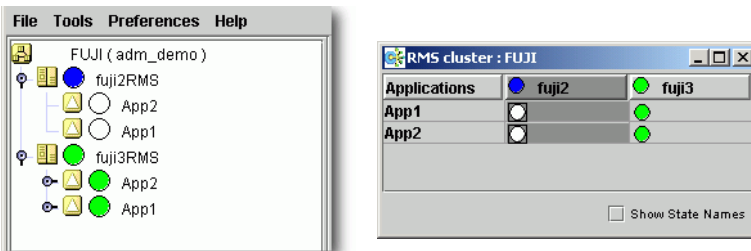


Figure 146: RMS main view and clusterwide table after shutdown on one node

6 Scalable controllers

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

- “Controller overview” on page 169
- “Scalable controllers and applications” on page 169
- “Online/offline processing and state transitions” on page 172

6.1 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.

6.2 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.

6.2.1 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.

6.2.2 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.

6.2.3 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, the Wizard Tools may restrict the choices available during configuration according to the application wizard employed.

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

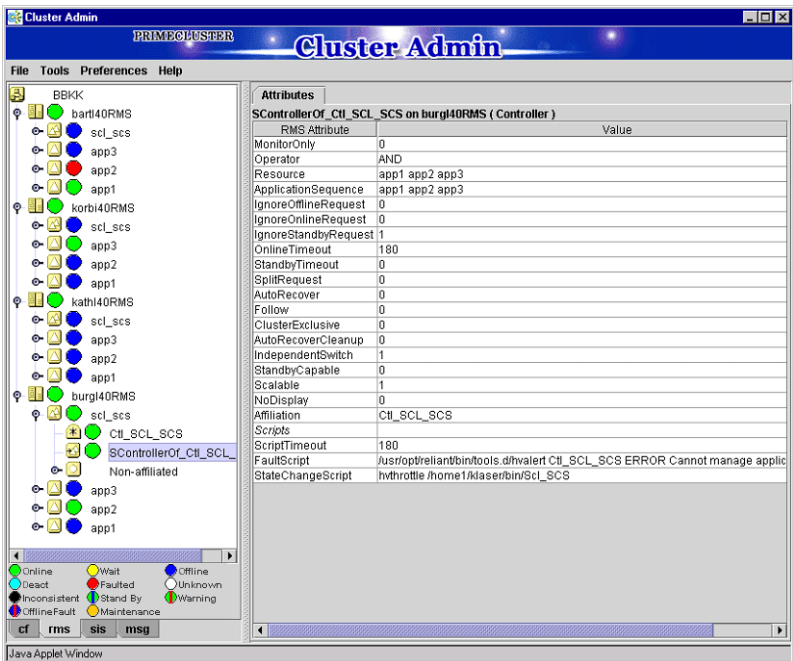


Figure 147: Scalable controller attributes

Some of the scalable controller attributes shown in the figure above can be adjusted in the 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 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.

6.3 Online/offline processing and state transitions

6.3.1 How controller states depend on controlled application states

Table 6 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 6: 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.

6.3.2 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 the offline request to each child application as a part of its switchover processing.

6.3.3 Controller warning state

A scalable controller may exist in a `Warning` state. This state will be posted for an online controller when at least one of its controlled applications is not online, or for a standby controller when at least one of its controlled applications is not standby. In this case, the actual controller state is online or standby, but it will post the state `Warning` for GUI and `hvdisp` to display. This alerts the administrator that the scalable parent application is not running in its full capacity.

Note that a scalable controller posts its `Warning` state only when controlled applications “degrade”; that is, at least one goes from online to offline or standby to faulted while the controller is online. When a controller is truly offline or faulted, it does not post a `Warning` state. Instead, it posts its true offline or faulted state.

6.3.4 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 or faulted, or when all controlled applications transition either to online or standby.

6.3.5 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 section "Script execution environment variables" on page 242.

The following variables are specific to state change scripts.

- **HV_REQUESTING_CONTROLLER**

If non-empty, contains the name of the controller and node that initiated the request for the current script execution.

- **HV_SCALABLE_CONTROLLER**

The name of the scalable controller that controls this application.

- **HV_SCALABLE_INFO**

Contains the list of all scalable applications everywhere in the cluster.

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 `Failed`. 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.

Alternatively, the script can use the RMS utility `hvthrottle` to serialize execution of the state change script with other scripts or with itself. Usage of `hvthrottle` is described in the online manual pages.

6.3.6 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.

6.3.7 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 `hvs switch -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.

6.3.8 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.

6.3.9 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>.
```

6.3.10 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 fuji2 and fuji3.
- App2 controls App1 with a follow mode controller.
- App3 controls App2 with a scalable mode controller.

Figure 148 shows the RMS configuration tree and the corresponding graph for this scenario.

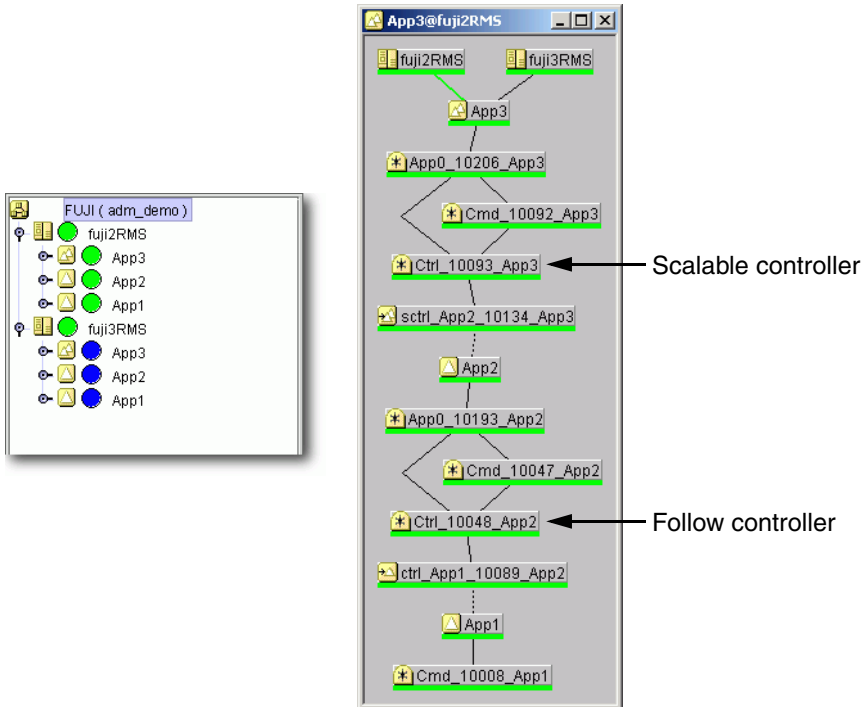


Figure 148: Maintenance mode controller example

6.3.10.1 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 `hvs` switch, either manually or automatically (Figure 149). Note that putting `App2` into maintenance mode automatically puts its follow-mode child application, `App1`, into maintenance mode as well.

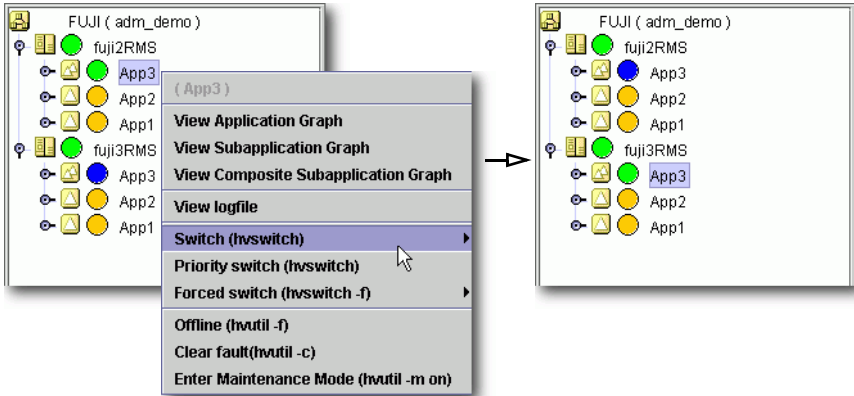


Figure 149: Switching scalable parent with child in maintenance mode

Likewise, if a scalable-mode parent application is put into maintenance mode, the child can still be switched between nodes with `hvswitch` (Figure 150). 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.

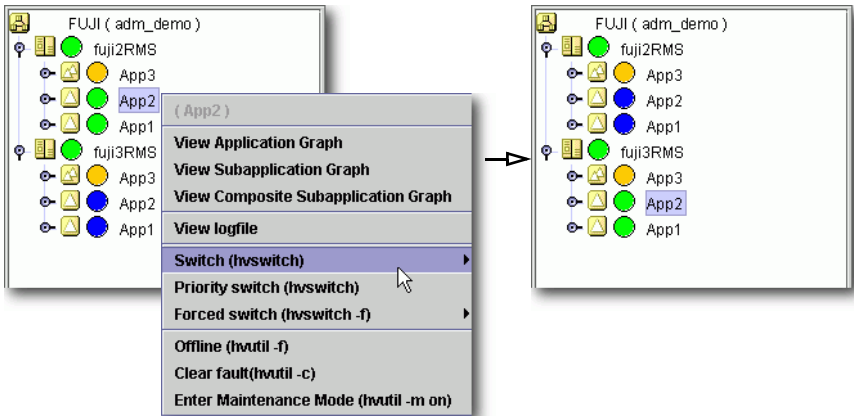


Figure 150: Switching scalable child with parent in maintenance mode

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

6.3.10.2 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 151).

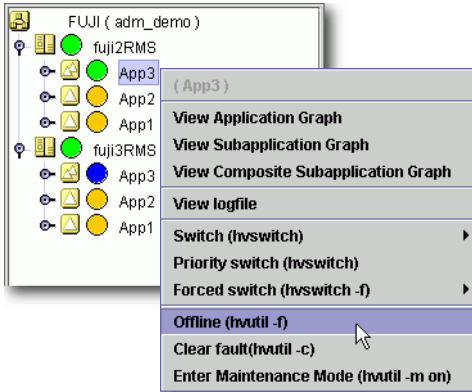


Figure 151: Attempting to take a parent offline with the child in maintenance mode

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 152) 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.

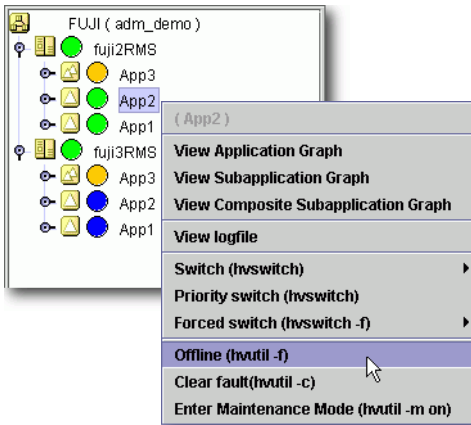


Figure 152: Attempting to take a child offline with the parent in maintenance mode

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.

6.3.10.3 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.

7 Advanced RMS concepts

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. It includes the following sections:

- “Internal organization” on page 185
- “Initializing” on page 186
- “Online processing” on page 191
- “Offline processing” on page 198
- “Fault processing” on page 202
- “Switch processing” on page 211
- “Special states” on page 213

7.1 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 were introduced in the chapter “Introduction” on page 9. The following sections provide more details about RMS internal structure and inter-object communication.

7.1.1 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.

7.1.2 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).

7.2 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 since they do not contain 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.

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 a object exits the `Unknown` state, it does not return to that state. An exception is only that `hvrreset` has been invoked. This command re-initializes the entire tree, the objects are forced back into the `Unknown` state and repeat the initialization steps.

The following RMS procedure examples 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.
- 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 153.

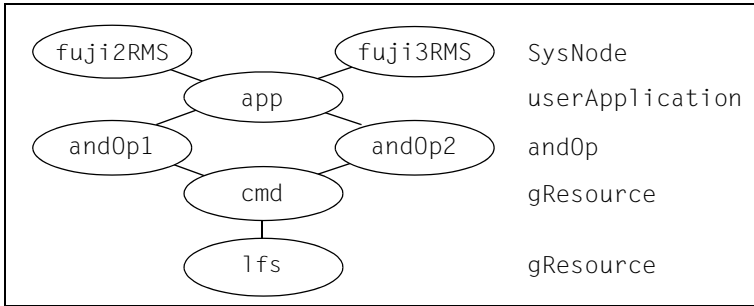


Figure 153: Object hierarchy for initializing examples

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 RMS graph for this configuration as produced by the Wizard Tools is shown in Figure 154. Note the line connecting *fuj i 2RMS* to *app* is green, indicating where the application is online.

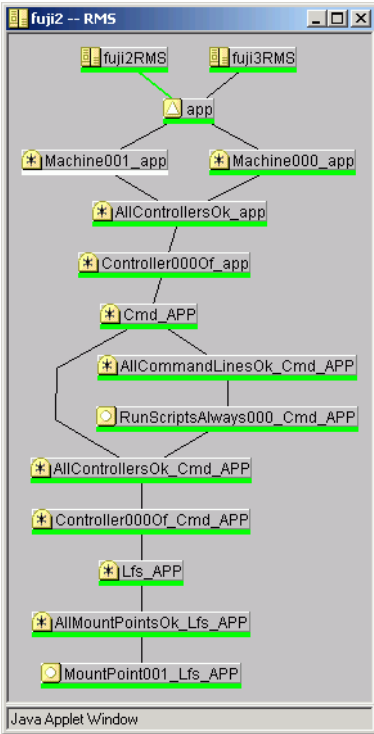


Figure 154: System graph for initializing examples—Wizard Tools

Figure 155 shows the corresponding output from 'hvdisp -a'.

```
# hvdisp -a

Local System: fuji2RMS
Configuration: /opt/SMAW/SMAWRrms/build/hvw_example.us

Resource          Type      HostName      State      StateDetails
-----
fuji3RMS          SysNode           Online
fuji2RMS          SysNode           Online
app               userApp           Online
Machine001_app   and0p    fuji3RMS
Machine000_app   and0p    fuji2RMS           Online
RunScriptsAlways000_Cmd_APP gRes                Online
MountPoint001_Lfs_APP gRes                Online
```

Figure 155: hvdisp output for initializing examples—Wizard Tools


The graphs contains additional connector objects and dependencies that are automatically inserted to ensure proper operation. Also, the object names automatically generated by the Wizard Tools are more complex than the simplified, generic names shown in the abstract graph of Figure 153. Neither the additional objects nor the complex names are required for a basic understanding of RMS operation. Therefore, to simplify the discussion, the following examples focus on the abstract graph and use the generic names.

Example 1

The following process for the configuration illustrated in Figure 153 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 a 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.

7.3 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.

7.3.1 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 section “Switch processing” on page 211).

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

7.3.1.1 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, left-click on an application (a pop-up menu is displayed).
2. Right-click on the *switch* or *online* selections within the pop-up 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.

7.3.1.2 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`.

7.3.2 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).

7.3.3 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 a 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.



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” on page 15.

Example 2

The scenario for this example is as follows:

- `AutoStartUp` attribute is set to 1.
- 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 1.
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 a 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.

7.3.4 Unexpected reports during online processing

Unexpected reports are detector reports that arrive out of sequence while a previous request is being processed. This could occur, for example, when RMS receives an `Offline` report while it is performing online processing. An unexpected report reflects an interim state that may be ignored under certain conditions as described below.

During online processing, unexpected reports are ignored from the point where the pre-online request for an object was received until the object's `OnlineScript` is finished, provided one of the following conditions is true at the end of processing:

- The “expected report,” e.g., `DetReportsOnline`, was received.
- The `OnlineScript` failed.
- The `ScriptTimeout` was exceeded without receiving an expected report.

7.3.5 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 section “Fault processing” on page 202). The following can cause faults during online processing:

- Detector reports the `Faulted` state.
- Detector reports the `Offline` state for a object that was reported as `Online`.
- Script fails with an exit status other than 0.
- Script fails with a timeout.
- Detector does not detect the object as online within a specific period after the `OnlineScript` completes.

7.3.6 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 section “Online processing” on page 191 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 a 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.

7.4 Offline processing

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

7.4.1 Offline request

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

- Manual offline request using the GUI or CLI (`hvvutil -f`)
- Manual switch request using the GUI or CLI (`hvs witch`)

- 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 section “Fault processing” on page 202). 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'`.

7.4.2 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 a 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`.

7.4.3 Unexpected reports during offline processing

RMS handling of unexpected reports during offline processing is similar to the online processing description earlier in this chapter.

During offline processing, unexpected reports are ignored from the point where the pre-offline request for an object was received until the object's `OfflineScript` is finished, provided one of the following conditions is true at the end of processing:

- The “expected report,” e.g., `DetReportsOffline`, was received.
- The `OfflineScript` failed.
- The `ScriptTimeout` was exceeded without receiving an expected report.

7.4.4 Fault situations during offline processing

The section “Fault processing” on page 202 describes the processing of any faults that occur during offline processing. The following can cause faults during offline processing:

- Detector reports the `Failed` state.
- Detector reports the `Online` state for an object after that object's `Offline` script completed successfully.
- Script fails with an exit status other than 0.
- Script fails with a timeout.
- Object is not reported by the detector as being `Offline` within the configured timeout period after the offline script completes.
- Child of a object indicates a fault.

7.4.5 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.

7.4.6 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.” A 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.

7.5 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.

7.5.1 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 section “Switch processing” on page 211). However, the `userApplication` is now in the `Failed` 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 `Failed` 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 `halt` 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.



A double fault causes the node to be eliminated if the application's `halt` attribute is set and the application can be switched to another 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 chapter “Appendix—Attributes” on page 219.

7.5.2 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.

7.5.3 AutoRecover attribute

An object of the type `gResource` that represents a local file system is one example of a 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.

7.5.4 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.

7.5.5 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 `Resource-Failure` and is online on node `fuji2RMS`.
- 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 `fuji3RMS`; all local objects on `fuji2RMS` 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.

7.5.6 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. If the graph has an `orOp` that was also initially online, the online processing will treat that `orOp` as a leaf object (the end of its branch). Objects below the `orOp` may continue to be in the faulted state as long as at least one of the children of the `orOp` is online. To initiate online or offline processing for the entire tree, use ‘`hvswitch -f`’ or ‘`hvutil -f`’ as described below.

2. The following command makes a forced online request:

```
hvswitch -f userApplication target_node
```

This starts online processing for the application on the specified node. If the command completes successfully, the application and every object in its graph (including those in `orOp` subtrees) is switched to the online state, and the fault is cleared.

3. 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 (including those in `orOp` subtrees) is 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.

7.5.7 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 the *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` setting 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 `fuji3RMS` starts online processing.

7.5.7.1 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 the *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 chapter “Appendix—Attributes” on page 219.

7.6 Switch processing

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

7.6.1 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 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.

7.7 Special states

7.7.1 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 156 shows the states of these objects if App1 is put into maintenance mode.

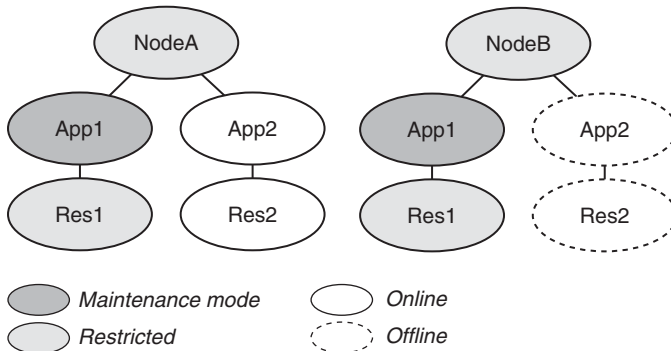


Figure 156: Example of maintenance mode restrictions

This simple example illustrates the following features:

- Maintenance mode for an application applies everywhere the application could be brought online. Therefore, the Maintenance state of App1 is the same on every node where it could run, no matter what the previous state

was on any given node. (For each node, the current **intended state**—the target state for App1 when it comes out of maintenance mode—is available in the application's *StateDetails* parameter in the GUI and CLI status 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 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.
- There may be other nodes in the cluster. As long as they are not included in the node list configured for App1, they are not affected by App1's maintenance mode.

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

7.7.2 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` 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`, this is not a true state in the RMS state machine: the true state is either `Offline` or `Faulted`. For most operations, the behavior of an `Inconsistent` `userApplication` is determined by the underlying true state. For instance, if the true state is `Offline`, and an `Offline` request is issued, no `Offline` script will be fired (see the section “Object is already in Offline state” on page 202).

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.

8 Appendix—Object types

The following alphabetical list describes all object types that are supplied with RMS and configured by PCS.

- **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.

- **Controller**

Required attributes:

- `Resource`

Object that allows a `userApplication` to control one or more `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`.

9 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 list all attributes along with their possible settings and default values.

9.1 Attributes available to the user

Attributes in this section can be changed using the PCS or Wizard Tools user interface.

- **AlternateIp**

Possible Values: Any interconnect name

Default: "" (empty)

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.

- **AutoRecover**

Possible Values: 0, 1

Default: 0

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.

- **AutoStartUp**

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. Automatically brings an application `Online` on the `SysNode` with the highest priority when RMS is started. Set to either 0 for no or 1 for yes.

You can override the `AutoStartUp` attribute for all `userApplication` objects by setting the `HV_AUTOSTARTUP` variable. See the description of `HV_AUTOSTARTUP` in the section “Local environment variables” on page 238.

- **AutoSwitchOver**

Possible Values: Valid string containing one or more of the following: `No`, `HostFailure`, `ResourceFailure`, `ShutDown`

Default: `No`

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 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`.

- **ClusterExclusive**

Possible Values: 0, 1

Default: 0

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.

The user can modify this attribute for a `cmdline` subapplication only. The configuration tools control this attribute for all other subapplications.

- **FaultScript**

Possible Values: Valid script (character)

Default: "" (empty)

Valid for all object types. Specifies a script to be run if the associated resource enters the `Faulted` state.

- **Halt**

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. Eliminates a node if a double fault occurs and the `userApplication` can be switched to another node.

- **I_List**

Possible Values: Space-separated list of SysNode names

Default: "" (empty)

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.

- **MaxControllers**

Possible Values: 0–512

Default: 512

Valid for userApplication objects. Upper limit of parent userApplication objects for the specified child application.

- **MonitorOnly**

Possible Values: 0, 1

Default: 0

Valid for resource objects. If set to 1, the state of the object is ignored by the parent when calculating the parent's state. Any parent should have at least one child for which MonitorOnly is not set.

- **OfflineScript**

Possible Values: Valid script (character)

Default: "" (empty)

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

Default: 0

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 may be used to establish consistent time across the nodes in the cluster. Refer to the manual page for `xntpd` 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)

Default: "" (empty)

Valid for all objects except `SysNode` objects. Specifies the script to bring the associated resource to the `Online` state.

- **PartialCluster**

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. Specifies if an application can negotiate online requests.

If set to 0, then the application can negotiate its online request only when all 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.

Note that a local `userApplication` that has its `PartialCluster` attribute set will not be affected by startup timeouts from remote nodes: the application can still go online on the local node. See the description of `HV_AUTOSTARTUP_WAIT` in the section “Global environment variables” on page 232.

- **PersistentFault**

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. If set to 1, the application maintains a `Faulted` state across an RMS shutdown and restart. The application returns to the `Faulted` state if it was `Faulted` before, unless the fault is explicitly cleared by either `hvutil -c` or `hvswitch -f`, or if RMS is restarted with the `Faulted SysNode` removed from the configuration.

- **PostOfflineScript**

Possible Values: Valid script (character)

Default: "" (empty)

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)

Default: "" (empty)

Valid for all objects except `SysNode` objects. Specifies the script to be run after the state of the associated resource changes to `Online`.

- **PreOfflineScript**

Possible Values: Valid script (character)

Default: "" (empty)

Valid for all objects except `SysNode` objects. Specifies the script to run before the object is taken to the `Offline` state.

- **PreOnlineScript**

Possible Values: Valid script (character)

Default: "" (empty)

Valid for all objects except `SysNode` objects. Specifies the script to be run before the associated resource is taken to the `Online` state.

- **PreserveState**

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. Specifies that resources are not to be taken `Offline` after a fault. Ignored if `AutoSwitchOver` is not set to `No`.

- **PriorityList**

Possible Values: Valid list of `SysNode` names (character)

Default: "" (empty)

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 a `Follow` controller, the order of nodes in `PriorityList` is ignored. However, each child application must be able to run on the nodes specified for the controller object.

- **ScriptTimeout**

Possible Values: 0–`MAXINT` (in seconds) or valid string of the form "timeout_value[:[offline_value][:online_value]]"

Default: 300

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–MAXINT

Default: 0

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

Default: 0

Valid for resource objects. If set to 1, the object performs standby processing on all 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 (|)

Default: "" (empty)

Valid for `userApplication` objects. The value specifies when the application will receive a standby request:

- *StartUp*—at startup, unless the application is already online or unless it is forced to go online due to the `AutoStartUp` attribute.
- *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)

Default: "" (empty)

Valid for resource objects with detector. Specifies the script to be run after the posted state of the associated resource changes to `Warning`.

9.2 Attributes managed by configuration wizards

Attributes in this section are managed internally by the configuration wizards or by RMS at runtime.

- **Affiliation**

Possible Values: Any string

Default: "" (empty)

Valid for resource objects. Used for display purposes in the user interface—no functional meaning within RMS.

- **AutoRecoverCleanup**

Possible Values: 0, 1

Default: 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

Default: Default type defined in the chapter “Appendix—Object types” on page 217.

Valid for all objects except `SysNode`. Describes the class of the resource object. Used by other programs for various purposes (for example, SNMP agent). This value is supplied by the configuration wizards.

- **Comment**

Possible Values: any string

Default: "" (empty)

Valid for all objects. Used for documentation in the configuration file—no functional meaning within RMS.

- **ControlledShutdown**

Possible Values: 0, 1

Default: 0

Valid for controlled `userApplication` objects. If set to 1, RMS does not send an `Offline` request to this application because an explicit request will be generated by a parent application during its offline processing.

- **ControlledSwitch**

Possible Values: 0, 1

Default: 0

Valid for controlled `userApplication` objects. If set to 1, the application is the child of a controller.

- **DetectorStartScript**

Possible Values: Any valid detector start script

Default: "" (empty)

Valid for resource object with detector. Specify the detector start command directly in the `<configname>.us` file.

- **HostName**

Possible Values: Any `SysNode` name

Default: "" (empty)

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.

- **IgnoreStandbyRequest**

Possible Values: 0, 1

Default: 1

Valid for `controller` objects. If set to 1, then neither `PreOnline` nor `Online` requests during standby processing will be propagated to the child application. If 0, then requests will be propagated. If the controller is not standby capable, then `IgnoreStandbyRequest` must be set to 1.

- **LastDetectorReport**

Possible Values: Online, Offline, Faulted, Standby

Default: (none)

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.

- **LieOffline**

Possible Values: 0, 1

Default: 1

Valid for all resource objects. If set to 1, allows the resource to remain `Online` during `Offline` processing.

- **NoDisplay**

Possible Values: 0, 1

Default: 0

Valid for all object types. If set to 1, specifies that the resource should not be displayed when `hvdisp` is active. Can be overridden by `'hvdisp -S'`.

- **NullDetector**

Possible Values: on, off

Default: off

Valid for resource objects with detector. Used to disable a detector at runtime by setting `NullDetector` to on. This attribute is for the use with dynamic reconfiguration only. `NullDetector` must never be set hard-coded to on in the RMS configuration file.

- **OfflineDoneScript**

Possible Values: Valid script (character)

Default: "" (empty)

Valid for `userApplication` objects. The last script run after the application has completed offline processing.

- **OnlineTimeout**

Possible Values: 0–MAXINT

Default: 0

Valid for `controller` objects. Specifies the time (in seconds) allowed for a controller not to react while a child application leaves the `Online` state.

- **PreCheckScript**

Possible Values: Valid script (character)

Default: "" (empty)

Valid for `userApplication` objects. Specifies the script to be forked as the first action during `Online` or `Standby` processing. If the script returns with a zero exit code, processing proceeds. If the script returns with an exit code other than zero, processing is not performed and an appropriate warning is logged to the `switchlog` file.

- **Resource**

Possible Values: Valid name (character)

Default: "" (empty)

Valid for `controller` objects. One or more names of child applications, separated by spaces and/or tabs.

- **rKind**

Possible Values: 0–2047

Default: none

Valid for `gResource` objects. Specifies the kind of detector for the object.

- **rName**

Possible Values: Valid string (character)

Default: none

Valid for `gResource` objects. Specifies a string to be forwarded to the generic detector.

- **SplitRequest**

Possible Values: 0, 1

Default: 0

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.

10 Appendix—Environment variables

This appendix provides a complete list of the environment variables used by RMS, grouped into the following types:

- “Global environment variables” on page 232
- “Local environment variables” on page 238
- “Script execution environment variables” on page 242

The discussion in the section “Environment variables” on page 34 describes how RMS manages environment variables.

10.1 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 `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`

10.2 Global environment variables

i 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.

i 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_AUTOSTARTUP_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.

Note that `HV_AUTOSTARTUP_WAIT` timeouts from remote nodes will not affect a local `userApplication` that has its `PartialCluster` attribute set: the application can still go online on the local node. See the description of the `PartialCluster` attribute in the section “Attributes available to the user” on page 219.



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_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 `'hvcm -h <timeout>'` 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.

- **HV_LOG_ACTION_THRESHOLD**

Possible values: 0–100

Default: 98

Defines the behavior of `hvllogcontrol`. If the used space on the log disk is larger or equal to this threshold, all subdirectories below `log` will be removed. Furthermore, if `HV_LOG_ACTION` is set to `on` and all subdirectories have already been removed, the actual log files will be removed too. Refer to the section “Local environment variables” on page 238 for more information on `HV_LOG_ACTION`.

- **HV_LOG_WARN_THRESHOLD**

Possible values: 0–100

Default: 95

Defines the behavior of `hvllogcontrol`. If the used space on the file system containing the RMS log disk is larger or equal to this threshold value, the `hvllogcontrol` script issues a warning to the user regarding the large amount of log files.

- **HV_LOH_INTERVAL**

Possible values: 0–`MAXINT`

Default: 30

Minimum difference in seconds when comparing timestamps to determine the last online host for an application. The last online host (LOH) specifies the host where the `userApplication` was online most recently. It is determined if the `OnlinePriority` attribute is set.

If the LOH timestamp entries of the `userApplication` on two hosts differ by less than this time interval, 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_WAIT_CONFIG**

Possible values: 0–MAXINT

Default: 120 (seconds)

Interval in seconds during which RMS waits to receive a configuration from an online host if RMS starts up as 'hvcm -C'. If the configuration is not received within *HV_WAIT_CONFIG* seconds, the local monitor will attempt to run with the configuration file specified in *RELIANT_BUILD_PATH*. If such a file does not exist, the local monitor will continue to run with the minimal configuration; in this case it is possible for it to join an already running RMS cluster via `hvjoin`.

- **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 log files which are older than 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, PCS, and Wizard Tools log files are stored.

- **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: 900 (seconds)

Defines the period (in seconds) that the command `hvs shut` waits before timing out and generating an error message. This value should be no less than the maximum time required by any application in the configuration to go offline on any node in the cluster.

If this value is too low, RMS terminates ungracefully: all running scripts are terminated immediately, and some resources under control of RMS will be left in an arbitrary state. These resources must be manually shut down before RMS can be restarted.

The default value will be adequate for some configurations, but each configuration must be considered individually. Long delays in offline processing may be caused by recurring issues such as large numbers of nodes or resources, or slow network connections or hardware. We recommend that you obtain the advice of an expert who is familiar with the applications and resources in your cluster.

If expert advice is unavailable, you can still estimate a reasonable value for `RELIANT_SHUT_MIN_WAIT`. Temporarily set the variable to a large value (e.g., 4000), run a series of tests that simulate production conditions, and then use the worst-case offline processing time plus a safety factor (e.g., 10%).



Due to the serious effects, you should diagnose the cause of an offline processing timeout before making another attempt to shut down RMS automatically.

10.3 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 section “Attributes available to the user” on page 219). 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: 0–*MAXINT*

Default: 0 (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.

- **HV_LOG_ACTION**

Possible values: on, off

Default: off

Determines if the current log files in the directory `RELIANT_LOG_PATH` are deleted if the used space on the file system is larger or equal to

`HV_LOG_ACTION_THRESHOLD`.

- **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–fork limit

Default: 30

Defines the maximum number of scripts RMS can have forked at any 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: 0–99

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.

- **HV_SCRIPTS_DEBUG**

Possible values: 0, 1

Default: 0

Controls debugging output from RMS scripts. If this variable is set to 1, each script writes detailed 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 PRIME-CLUSTER products. To disable script debug message logging, delete the `HV_SCRIPTS_DEBUG` entry or set `HV_SCRIPTS_DEBUG=0` in `hvenv.local`.

- **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`.

- **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: `fuji2RMS`), 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 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.

10.4 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, Faulted, NoReport

Last detector report for the current object.

- **HV_OFFLINE_REASON**

Possible values: one of DEACT, SWITCH, FAULT, STOP

Reason for ongoing offline processing:

DEACT: deact request ('hvutil -d')

SWITCH: manual switchover ('hvswitch')

FAULT: follow-up processing after a previous resource failure

STOP: userApplication is stopped ('hvutil -f', 'hvutil -c', 'hvshut').

- **HV_NODENAME**

Possible values: any object name

Name of current object.

- **HV_SCRIPT_TYPE**

Possible values: one of PreCheckScript, PreOnlineScript, OnlineScript, PostOnlineScript, PreOfflineScript, OfflineScript, PostOfflineScript, OfflineDoneScript, FaultScript

Script type.

- **NODE_SCRIPTS_TIME_OUT**

Possible values: 0–MAXINT

Timeout value for the current object and script type.

11 Appendix—List of manual pages

This appendix lists the online manual pages for CCBR, CF, CFS, CIP, Monitoring Agent, PAS, RCVN, Resource Database, RMS, RMS Wizards, SCON, SF, SIS, and Web-Based Admin View.

To display a manual page, type the following command:

```
$ man man_page_name
```

11.1 CCBR

System administration

`cfbackup`

save the cluster configuration information for a PRIMECLUSTER node

`cfrestore`

restore saved cluster configuration formation on a PRIMECLUSTER node

11.2 CF

System administration

`cfconfig`

configure or unconfigure a node for a PRIMECLUSTER cluster

`cfset`

apply or modify `/etc/default/cluster.config` entries into the CF module

`cfrecon`

dynamically reconfigure the cluster interconnects used by a node

`cftool`

print node communications status for a node or the cluster

`rcqconfig`

configures or reports cluster quorum settings

11.3 CFS

- fsck_rcfs
file system consistency check and interactive repair
- mount_rcfs
mount RCFS file systems
- rcfs_fumount
force unmount RCFS mounted file system
- rcfs_list
list status of RCFS mounted file systems
- rcfs_switch
manual switchover or failover of a RCFS file system
- ngadmin
node group administration utility
- cfsmntd
cfs mount daemon for RCFS

11.4 CIP

System administration

- cipconfig
start or stop CIP 2.0
- ciptool
retrieve CIP information about local and remote nodes in the cluster

File format

- cip.cf
CIP configuration file format

11.5 Monitoring Agent

System administration

`cldevparam`

Changes and displays the tunable operation environment for asynchronous monitoring

`clmmbmonctl`

Starts, stops, restarts, and displays the operating system of the MMB asynchronous monitoring daemon

`clmmbsetup`

Registers, changes, deletes, and displays MMB information

`clrcimonctl`

Start, stop or restart of the RCI monitoring agent daemon, and display of daemon presence

`clrccumonctl`

Start, stop or restart of the console monitoring agent daemon, and display of daemon presence

`clrccusetup`

Registers, changes, deletes, or displays console information

11.6 PAS

System administration

`mipcstat`

MIPC statistics

`clmstat`

CLM statistics

11.7 RCVM

RCVM is not available in all markets.

System administration

`dkconfig`
virtual disk configuration utility

`dkmigrate`
virtual disk migration utility

`vdisk`
virtual disk driver

`dkmirror`
mirror disk administrative utility

File format

`dktab`
virtual disk configuration file

11.8 Resource Database



To display a Resource Database manual page, add `/etc/opt/FJSCluster/man` to the environment variable `MANPATH`.

System administration

`clautoconfig`
execute of the automatic resource registration

`clbackuprdb`
save the resource database

`clexec`
execute the remote command

`cldeldevice`
delete resource registered by automatic resource registration

`clinitreset`
reset the resource database

`clrestorerdb`
restore the resource database

`clsetparam`
display and change the resource database operational environment

`clsetup`
set up the resource database

`clstarttrsc`
resource activation

`clstoptrsc`
resource deactivation

`clsyncfile`
distribute a file between cluster nodes

User command

`clgettree`
display the tree information of the resource database

11.9 RMS

System administration

`hvassert`
assert (test for) an RMS resource state

`hvattrib`
make cluster-wide attribute changes at runtime from a single node
(installed with PCS or the Wizard Tools)

`hvcm`
start the RMS configuration monitor

`hvconfig`
display or save the RMS configuration file

`hvdisp`
display RMS resource information

`hvdistrib`
distribute RMS configuration files

- hvdump
collect debugging information about RMS
- hvgdmake
compile an RMS custom detector
- hvlogclean
clean RMS log files
- hvrclv
change default RMS start run level
- hvreset
reinitialize the graph of an RMS user application (for use by experts in test conditions only—not for use on production clusters)
- hvsetenv
controls automatic startup of RMS or all user applications on the local host
- hvshut
shut down RMS
- hvswitch
switch control of an RMS user application resource to another node
- hvthrottle
prevent multiple RMS scripts from running simultaneously
- hvutil
manipulate availability of an RMS resource

File formats

- hvenv.local
RMS local environment configuration file

11.10 RMS Wizards

RMS Wizard Tools and RMS Wizard Kit

RMS Wizards are documented as HTML pages in the SMAWRhv-do package on the CD-ROM. After installing this package, the documentation is available in the following directory:

<RELIANT_PATH>/htdocs.solaris/wizards.en (Solaris)

<RELIANT_PATH>/htdocs.linux/wizards.en (Linux)

The default value of <RELIANT_PATH> is /opt/SMAW/SMAWRms/.

11.11 SCON

scon

start the cluster console software

11.12 SF

System administration

rcsd

shutdown daemon for the Shutdown Facility

sdtool

interface tool for the shutdown daemon

File formats

rcsd.cfg

configuration file for the Shutdown Daemon

SA_blade.cfg

configuration file for FSC server blade Shutdown Agent

SA_rccu.cfg

configuration file for RCCU Shutdown Agent

SA_rps.cfg

configuration file for Remote Power Switch Shutdown Agent

- SA_rsb.cfg
configuration file for RemoteView Services Board Shutdown Agent
- SA_scon.cfg
configuration file for SCON Shutdown Agent
- SA_pprci.cfg
configuration file for RCI Shutdown Agent (PRIMEPOWER only)
- SA_sspint.cfg
configuration file for Sun E10000 Shutdown Agent
- SA_sunF.cfg
configuration file for sunF system controller Shutdown Agent
- SA_wtinps.cfg
configuration file for WTI NPS Shutdown Agent

11.13 SIS

System administration

- dtcpadmin
start the SIS administration utility
- dtcpd
start the SIS daemon for configuring VIPs
- dtcpstat
status information about SIS

11.14 Web-Based Admin View

System administration

- fjsvwvbs
stop Web-Based Admin View
- fjsvwvcnf
start, stop, or restart the web server for Web-Based Admin View
- wvCntl
start, stop, or get debugging information for Web-Based Admin View

- wvGetparam
display Web-Based Admin View's environment variable
- wvSetparam
set Web-Based Admin View environment variable
- wvstat
display the operating status of Web-Based Admin View

Glossary

Items in this glossary that apply to specific PRIMECLUSTER products are indicated with the following notation:

- (CF)—Cluster Foundation
- (PCS)—PRIMECLUSTER Configuration Services
- (RMS)—Reliant Monitor Services
- (RCVM)—Volume Manager (not available in all markets)
- (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.

activating a configuration (RMS)

Preparing an RMS configuration to be run on a cluster. This involves two major actions: first, the configuration is **generated** on the host where the configuration was created or edited; second, the configuration is **distributed** to all nodes affected by the configuration. The user can activate a configuration using PCS, the Wizard Tools, or the CLI.

See also *generating a configuration (RMS)*, *distributing a configuration (RMS)*.

Administrative LAN

In PRIMECLUSTER configurations, an Administrative LAN is a private local area network (LAN) on which machines such as the System Console and Cluster Console reside. Because normal users do not have access to the Administrative LAN, it provides an extra level of security. The use of an Administrative LAN is optional.

See also *public LAN*.

API

See *Application Program Interface*.

application (RMS)

A resource categorized as a `userApplication` used to group resources into a logical collection.

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 PCS, the Wizard Tools, or the PCS 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 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

See *Cluster Configuration Backup and Restore*.

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, RMS, PCS, and SIS. Cluster Admin is available from the Web-Based Admin View interface.

See also *Cluster Foundation (CF)*, *Reliant Monitor Services (RMS)*, *PRIME-CLUSTER Configuration Services (PCS)*, *Scalable Internet Services (SIS)*, *Web-Based Admin View*.

Cluster Configuration Backup and Restore

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 PRIMECLUSTER communications.

Cluster Join Services (CF)

This PRIMECLUSTER module handles the forming of a new cluster and the addition of nodes.

concatenated virtual disk (RCVM)

Concatenated virtual disks consist of two or more pieces on one or more disk drives. They correspond to the sum of their parts. Unlike simple virtual disks where the disk is subdivided into small pieces, the individual disks or partitions are combined to form a single large logical disk.

See also *mirror virtual disk (RCVM)*, *simple virtual disk (RCVM)*, *striped virtual disk (RCVM)*, *virtual disk*.

Configuration Definition Language (PCS)

The syntax for PCS configuration templates.

See also *PRIMECLUSTER Configuration Services (PCS)*.

configuration file (RMS)

The RMS configuration 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 base monitor.

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)*.

distributing a configuration (RMS)

The process of copying a configuration file and all of its associated scripts and detectors to all nodes affected by the configuration. This is normally done automatically when the configuration is **activated** using PCS, the Wizard Tools, or the CLI.

See also *activating a configuration (RMS)*, *generating a configuration (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.

Glossary

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)*.

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)*.

GDS

See *Global Disk Services*.

generating a configuration (RMS)

The process of creating a single configuration file that can be distributed to all nodes affected by the configuration. This is normally done automatically when the configuration is **activated** using PCS, the RMS Wizards, or the CLI.

See also *activating a configuration (RMS)*, *distributing a configuration (RMS)*.

GFS

See *Global File Services*.

GLS

See *Global Link Services*.

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 employed to avoid single points of failure.

See also *Reliant Monitor Services (RMS)*.

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*.

JOIN (CF)

See *Cluster Join Services (CF)*.

keyword

A word that has special meaning in a programming language. For example, in the 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 base monitor, wizards, and detectors can have their own log files.

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.

Meta Data Server

GFS daemon that centrally manages the control information of a file system (meta-data).

mirrored disks (RCVM)

A set of disks that contain the same data. If one disk fails, the remaining disks of the set are still available, preventing an interruption in data availability.

See also *mirrored pieces (RCVM)*.

mirrored pieces (RCVM)

Physical pieces that together comprise a mirrored virtual disk. These pieces include mirrored disks and data disks.

See also *mirrored disks (RCVM)*.

mirror virtual disk (RCVM)

Mirror virtual disks consist of two or more physical devices, and all output operations are performed simultaneously on all of the devices.

See also *concatenated virtual disk (RCVM)*, *simple virtual disk (RCVM)*, *striped virtual disk (RCVM)*, *virtual disk*.

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.

node

A host which is a member of a cluster. A computer node is the same as a computer.

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)

In the configuration file or a system graph, this is a representation of a physical or virtual resource.

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. The keyword associated with an object definition is `object`.

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 configuration file or system graph that has at least one child.

See also *child (RMS)*, *configuration file (RMS)*, *system graph (RMS)*.

PCS

See *PRIMECLUSTER Configuration Services (PCS)*.

PCS Wizard Kit (PCS)

RMS configuration products that have been designed for specific applications. Each component of the PCS Wizard Kit includes customized default settings, subapplications, detectors, and scripts. These application wizards also tailor the PCS interface to provide controls for the additional features.

See also *PCS, Reliant Monitor Services (RMS)*.

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 Configuration Services (PCS)

The graphical configuration interface for PRIMECLUSTER products. PCS uses standard templates written in Configuration Definition Language (CDL) to provide a user-friendly configuration environment for products such as RMS. The standard templates can be modified or replaced to provide a customized interface for specific applications or installations.

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 Assigned Numbers Authority. They may be used internally by any organization but, because different organizations can use the same addresses, they should never be made visible to the public internet.

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*.

RC2000

Software (Java application) that provides functions for operating the operating system console of each node.

RCCU

Abbreviation for Remote Console Connection Unit.

See also *remote console connection unit*.

redundancy

This is the capability of one object to assume the resource load of any other object in a cluster, and the capability of RAID hardware and/or RAID software to replicate data stored on secondary storage devices.

Reliant Monitor Services (RMS)

The package that maintains high availability of user-specified resources by providing monitoring and switchover capabilities.

remote console connection unit

Device that converts an RS232C interface and a LAN interface. This device allows another device (personal computer) that is connected to the LAN to use the TTY console functions through the Telnet function.

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)*.

Glossary

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)

RMS configuration products that have been designed for specific applications. Each component of the Wizard Kit includes customized default settings, subapplications, detectors, and scripts. These application wizards also tailor the RMS Wizard Tools interface to provide controls for the additional features.

See also *RMS Wizard Tools (RMS)*, *Reliant Monitor Services (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*.

Scalable Internet Services (SIS)

Scalable Internet Services is a TCP connection load balancer, and dynamically balances network access loads across cluster nodes while maintaining normal client/server sessions for each connection.

scalability

The ability of a computing system to dynamically handle any increase in work load. Scalability is especially important for Internet-based applications where growth caused by Internet usage presents a scalable challenge.

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 Shutdown Facility provides the interface for managing the shutdown of cluster nodes when error conditions occur. The SF also cares for advising other PRIMECLUSTER products of the successful completion of node shutdown so that recovery operations can begin.

simple virtual disk (RCVM)

Simple virtual disks define either an area within a physical disk partition or an entire partition.

See also *concatenated virtual disk (RCVM)*, *mirror virtual disk (RCVM)*, *striped virtual disk (RCVM)*, *virtual disk*.

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)*.

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.

striped virtual disk (RCVM)

Striped virtual disks consist of two or more pieces. These can be physical partitions or further virtual disks (typically a mirror disk). Sequential I/O operations on the virtual disk can be converted to I/O operations on two or more physical disks. This corresponds to RAID Level 0 (RAID0).

See also *concatenated virtual disk (RCVM)*, *mirror virtual disk (RCVM)*, *simple virtual disk (RCVM)*, *virtual disk*.

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)*.

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 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

With virtual disks, a pseudo device driver is inserted between the highest level of the OS logical Input/Output (I/O) system and the physical device driver. This pseudo device driver then maps all logical I/O requests on physical disks.

See also *concatenated virtual disk (RCVM)*, *mirror virtual disk (RCVM)*, *simple virtual disk (RCVM)*, *virtual disk*.

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. An enabler is a type of wizard.

Glossary

Wizard Kit (RMS)

See *PCS Wizard Kit (PCS)*, *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

DLPI

Data Link Provider Interface

Abbreviations

ELM	Enhanced Lock Manager
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
JOIN	cluster join services module
LAN	local area network
MDS	Meta Data Server
MIB	Management Information Base
MIPC	Mesh Interprocessor Communication

MMB	Management Board
NIC	network interface card
NSM	Node State Monitor
OSD	operating system dependent
PAS	Parallel Application Services
PCS	PRIMECLUSTER Configuration Services
RCCU	Remote Console Connection Unit
RCFS	PRIMECLUSTER File Share
RCI	Remote Cabinet Interface
RCVM	PRIMECLUSTER Volume Manager
RMS	Reliant Monitor Services
SA	Shutdown Agent
SAN	Storage Area Network
SCON	single console software

Abbreviations

SD

Shutdown Daemon

SF

Shutdown Facility

SIS

Scalable Internet Services

VIP

Virtual Interface Provider

XSCF

eXtended System Control Facility

Figures

Figure 1:	Overview of PRIMECLUSTER products	10
Figure 2:	Interface between RMS and the operating system	15
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	21
Figure 8:	Controller structure with multiple children	22
Figure 9:	Relationship between RMS and RMS Wizards	24
Figure 10:	NFS Lock Failover screen	47
Figure 11:	Main configuration menu when RMS is not active	51
Figure 12:	Main configuration menu when RMS is running	55
Figure 13:	Application type selection	56
Figure 14:	Menu leading to basic settings	57
Figure 15:	Menu to configure basic settings	58
Figure 16:	Menu to configure non-basic settings	59
Figure 17:	Main configuration menu	60
Figure 18:	Activating a configuration	61
Figure 19:	Quitting the Main configuration menu	62
Figure 20:	Main configuration menu	68
Figure 21:	Add hosts to a cluster menu	69
Figure 22:	Remove hosts from a cluster menu	70
Figure 23:	Main configuration menu	71

Figures

Figure 24:	Application type selection menu	71
Figure 25:	Prompting for further actions	73
Figure 26:	Consistency check and Machines+Basics menu	74
Figure 27:	List of nodes for failover procedure	75
Figure 28:	Machines+Basics menu for additional nodes	75
Figure 29:	AutoSwitchOver mode	76
Figure 30:	Setting flags for AutoSwitchOver mode	76
Figure 31:	Saving settings	77
Figure 32:	Non-basic settings	78
Figure 33:	Prompting for display specification	79
Figure 34:	List of display options	80
Figure 35:	Successful consistency check for APP1	81
Figure 36:	Turnkey wizard DEMO	82
Figure 37:	Main configuration menu	83
Figure 38:	Successful configuration activation	83
Figure 39:	Quitting the Main configuration menu	84
Figure 40:	Starting again with the Main configuration menu	85
Figure 41:	Application type selection menu	86
Figure 42:	Prompting for further specification	87
Figure 43:	Machines+Basics menu	87
Figure 44:	List of nodes for failover procedure	88
Figure 45:	Machines+Basics menu	88
Figure 46:	Non-basic settings	89
Figure 47:	Assigning a controller	90
Figure 48:	List of applications to be chosen as controlled applications	90

Figure 49:	Menu for setting controller flags	91
Figure 50:	Changing controller timeout period	91
Figure 51:	Saving flags for controller	92
Figure 52:	Indication of flags set for controller	92
Figure 53:	Menu with settings for GENERIC turnkey wizard	93
Figure 54:	Main configuration menu	93
Figure 55:	Main configuration menu	94
Figure 56:	Activating the configuration for the second time	94
Figure 57:	Return to Main configuration menu	95
Figure 58:	Web-Based Admin View login	99
Figure 59:	Invoking the Cluster Services GUI	100
Figure 60:	Invoking Cluster Admin	100
Figure 61:	Cluster Admin initial connection menu	101
Figure 62:	Security certificate dialog	102
Figure 63:	Main Cluster Admin window—Initial view	103
Figure 64:	Main Cluster Admin window—message view	104
Figure 65:	Main Cluster Admin window—RMS view	105
Figure 66:	RMS tree with a controller object	107
Figure 67:	Command pop-up menu for a node	108
Figure 68:	Command pop-up menu for an online application	109
Figure 69:	Command pop-up menu for an offline application	109
Figure 70:	Confirmation pop-up window	110
Figure 71:	Confirmation pop-up window for scalable application	110
Figure 72:	Displaying global environment variables	110
Figure 73:	Global environment variable view	111
Figure 74:	Displaying local environment variables	112

Figures

Figure 75:	Local environment variables view	112
Figure 76:	Displaying application and object states	113
Figure 77:	Configuration information or object attributes	114
Figure 78:	Viewing the RMS switchlog file using a context menu	115
Figure 79:	Viewing the RMS switchlog file using the Tools menu	115
Figure 80:	Viewing an application log using a context menu	116
Figure 81:	Viewing an RMS log	116
Figure 82:	Viewing the RMS switchlog file in a detached window	117
Figure 83:	Search based on date and time range	118
Figure 84:	Search based on resource name	119
Figure 85:	Search based on severity level	120
Figure 86:	Search based on keyword	121
Figure 87:	Using the Find pop-up in log viewer	122
Figure 88:	Opening the clusterwide table	123
Figure 89:	Clusterwide table	123
Figure 90:	Clusterwide table with state names	123
Figure 91:	Faulted applications in the clusterwide table	124
Figure 92:	Offline applications in the clusterwide table	124
Figure 93:	Faulted and offline applications in the clusterwide table	125
Figure 94:	Split-brain conditions in the clusterwide table	125
Figure 95:	Using command pop-ups in clusterwide table	126
Figure 96:	Cluster state before RMS is shut down	127
Figure 97:	Cluster state after RMS restart with different configuration	127

Figure 98:	Starting RMS from the main menu	128
Figure 99:	RMS Start Menu for all nodes	129
Figure 100:	RMS Start Menu for individual nodes	129
Figure 101:	Starting RMS on individual nodes	130
Figure 102:	Controlling automatic RMS startup—step 1	131
Figure 103:	Controlling automatic RMS startup—step 2	132
Figure 104:	Using the Tools menu to stop RMS	133
Figure 105:	Stopping RMS on all available nodes	133
Figure 106:	Stopping RMS on one node from the list	134
Figure 107:	Using the context menu to stop RMS on one node	135
Figure 108:	Stopping RMS on one node	136
Figure 109:	Stopping RMS while keeping applications —confirmation	136
Figure 110:	Controlling automatic application startup—step 1	138
Figure 111:	Controlling automatic application startup—step 2	138
Figure 112:	Starting an application	140
Figure 113:	Switching an application	141
Figure 114:	Switching a busy application	142
Figure 115:	Shutting down an application	142
Figure 116:	Clearing an application fault	144
Figure 117:	Clearing an application fault—confirmation dialog	144
Figure 118:	Starting maintenance mode for all applications	146
Figure 119:	Starting maintenance mode for a single application	146
Figure 120:	Maintenance mode confirmation for all applications	147
Figure 121:	Maintenance mode confirmation for one application	147
Figure 122:	Typical cluster in maintenance mode	148

Figures

Figure 123:	Normal operation of independent application	149
Figure 124:	Normal maintenance mode exit for all applications	149
Figure 125:	Normal maintenance mode exit for a single application	150
Figure 126:	Forced maintenance mode confirmation for all applications	150
Figure 127:	Application with fault condition during maintenance mode	151
Figure 128:	Fault clearing from configuration tree	151
Figure 129:	Fault clearing from clusterwide table	151
Figure 130:	Application returned to normal maintenance mode	152
Figure 131:	Viewing the RMS full graph on a node	155
Figure 132:	Typical RMS full graph	155
Figure 133:	RMS full graph—object tooltip	156
Figure 134:	RMS full graph—object details	157
Figure 135:	Viewing an RMS application graph	158
Figure 136:	Typical RMS application graph	159
Figure 137:	Viewing an RMS subapplication graph	159
Figure 138:	Typical RMS subapplication graph	160
Figure 139:	Viewing an RMS composite subapplication graph	161
Figure 140:	Typical composite subapplication graph	162
Figure 141:	Using a command pop-up menu from the graph	163
Figure 142:	Displaying an RMS graph with affiliation names	164
Figure 143:	RMS graph with resource names	165
Figure 144:	RMS graph with resource and affiliation names	166
Figure 145:	RMS graph after shutdown on one node	167

Figure 146:	RMS main view and clusterwide table after shutdown on one node	167
Figure 147:	Scalable controller attributes	171
Figure 148:	Maintenance mode controller example	180
Figure 149:	Switching scalable parent with child in maintenance mode	181
Figure 150:	Switching scalable child with parent in maintenance mode	181
Figure 151:	Attempting to take a parent offline with the child in maintenance mode	182
Figure 152:	Attempting to take a child offline with the parent in maintenance mode	183
Figure 153:	Object hierarchy for initializing examples	188
Figure 154:	System graph for initializing examples—Wizard Tools	189
Figure 155:	hvdsp output for initializing examples—Wizard Tools	190
Figure 156:	Example of maintenance mode restrictions	213

Tables

Table 1: Available CLI commands	30
Table 2: RMS base directory structure	36
Table 3: Log directory structure	37
Table 4: RMS host name conventions in /etc/hosts	43
Table 5: RMS severity level description	120
Table 6: Dependence of scalable controller states on child applications	172

Index

- .rhosts 43
- /etc/cip.cf 75
- /etc/dfs/dfstab 45
- /etc/exports 47
- /etc/fstab 47
- /etc/hosts 42, 69
- /etc/inittab 32, 132
- /etc/nsswitch.conf 48
- /etc/vfstab 45
- /opt/SMAW/SMAWRrms 35, 36
- /opt/SMAW/SMAWRrms/build/ 68
- /opt/SMAW/SMAWRrms/etc/ 130
- /opt/SMAW/SMAWRrms/etc/
 CONFIG.rms 130
- /var/adm/messages 48
- /var/log/messages 48
- /var/opt/SMAWRrms/log 37
- /var/opt/SMAWRrms/log/ 117
- >> input prompt 50
- ~ as spaces 79

A

- activating
 - applications 143
 - configuration 53, 59, 83
 - configuration second time 94
 - configurations 128
- Adaptive Services
 - Tools menu 154
- administrative privileges 99
- Affiliation attribute 226
- affiliation names, graphs 166
- alternate interfaces 43
- Alternatelp attribute 219
- Alternatelps 43
- andOp objects 217
- applets, Java, trusted 101
- application logs
 - searching text 122
 - viewing 115

- applications
 - activating 143
 - as objects 14
 - clearing faults 144
 - deactivating 143
 - dependencies 106
 - displaying states 113
 - failover 16
 - going offline 198
 - maintenance mode,
 - clusterwide 147
 - objects 64
 - overriding AutoStartUp 238
 - starting 140
 - startup, overriding 137
 - stopping 136, 142
 - switching 141, 211
 - switching to Standby state 143
 - switching to SysNode 32
 - taking offline 142
 - viewing attributes 114
 - viewing graph 158
 - viewing log files 114
 - viewing logs 115
- Attach button 117
- attributes 34
 - Affiliation 226
 - Alternatelp 219
 - AutoRecover 219
 - AutoRecoverCleanup 226
 - AutoStartUp 219
 - AutoSwitchOver 220
 - Class 226
 - ClusterExclusive 220
 - Comment 227
 - ControlledShutdown 227
 - DetectorStartScript 227
 - ENV object 217
 - FaultScript 220
 - for andOp objects 217
 - for Controller objects 217

- for gResource objects 217
- for orOp objects 218
- for SysNode objects 218
- for userApplication objects 218
- Halt 220
- HostName 227
- I_List 221
- IgnoreStandbyRequest 228
- LastDetectorReport 228
- LieOffline 228
- MaxControllers 221
- MonitorOnly 221
- NoDisplay 228
- NullDetector 228
- OfflineDoneScript 229
- OfflineScript 221
- OnlinePriority 221
- OnlineScript 222
- OnlineTimeout 229
- PartialCluster 222
- PersistentFault 223
- PostOfflineScript 223
- PostOnlineScript 223
- PreCheckScript 229
- PreOfflineScript 223
- PreOnlineScript 224
- PreserveState 224
- PriorityList 224
- Resource 229
- rName 230
- ScriptTimeout 224
- ShutdownPriority 225
- SplitRequest 230
- StandbyCapable 225
 - viewing, for objects 114
- WarningScript 226
- automatic RMS startup 131
- AutoRecover attribute 219
 - fault processing 206
- AutoRecoverCleanup attribute 226
- AutoStartUp attribute 219
 - overriding 137, 238
- AutoSwitchOver attribute 220
 - fault processing 203

B

- base monitor 26
 - cluster monitoring 234
 - detectors 64
 - high availability 13
 - locking process in memory 239
- basic settings, wizards 57
- bin, RMS directory 36
- browser 98
- build, RMS directory 36
- buttons
 - Attach, log viewer 117
 - Detach, log viewer 117
 - Filter, log viewer 117

C

Caution

- defined* 6
- Do not explicitly set RMS environment variables in the user environment 231
- Do not modify the /bin/hvenv file 34
- 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 233
- Leaving applications running after stopping RMS can lead to data corruption 134
- Leaving applications running after stopping RMS or using forced shutdown can cause data corruption 135
- Use care when clearing a SysNode Wait state manually 145
- Use hvshut -A, -f, and -L options carefully as they could result in inconsistencies or data corruption 137

- Use the forced switch mode only if an application cannot be switched normally 141
- Use the hvswitch -f option carefully as it could result in inconsistencies or data corruption 140
- CF 12
 - Cluster Admin view 103
 - configuration 106
 - LEFTCLUSTER 210
 - node names 97
 - required for ELM 234
 - tabbed view, Cluster Admin 103
- CF commands
 - cfconfig 245
 - cfrecon 245
 - cftool 245
 - rcqconfig 245
- cfset command 245
- cfsmntd command 246
- changing
 - configurations, clusterwide table
 - display of 126
 - detail level in graphs 164
 - environment variables 231
 - HV_AUTOSTARTUP 138
 - HV_RCSTART 132
- CIP commands
 - cip.cf 246
 - cipconfig 246
 - ciptool 246
- Class attribute 226
- clbackuprdb 248
- clearing
 - faulted resources 33
 - faults 144
 - faults, in maintenance mode 150
 - hung nodes 33
 - SysNode Wait state 145
- clgettree 249
- CLI *see* RMS commands
- clinitreset 248
- closely coupled applications 19
- clrestorerdb command 249
- clsetparam command 249
- clsetup command 249
- clstartsrc command 249
- clstopsrc command 249
- cluster 1
 - high availability 10
 - monitoring by RMS 234
 - services 9
- Cluster Admin 26
 - administrative privileges 99
 - application graph 158
 - clusterwide table 123
 - command pop-up menus 108
 - graph 161
 - GUI 97
 - initial view 103
 - logging in 99
 - object attributes 114
 - operator privileges 99
 - overview 10
 - primary management server 98
 - RMS graphs 154
 - RMS tree 106
 - root privileges 99
 - searching log text 122
 - secondary management server 98
 - starting 98
 - switchlog 115
 - switchlog panel 117
 - SysNode selection 108
 - userApplication selection 108
 - using 98
 - viewing log files 114
- cluster file system 9
- Cluster Foundation *see* CF 12
- cluster node
 - detector timeout for remote 238
 - ignore at startup 232
 - wait to report online 233
- cluster volume management 9
- ClusterExclusive attribute 220
- clusterwide
 - maintenance mode 147

- clusterwide table 123
 - displaying state names 123
 - special display order 124
- Cmdline, resource wizard 41
- command pop-up menus
 - see also* context menus
 - RMS graph 163
 - RMS tree 108
- commands *see* RMS commands
- Comment attribute 227
- composite subapplication graph 161
- CONFIG.rms default startup
 - file 132, 239
- CONFIG.rms, default RMS startup file 130
- configuration tree
 - RMS 106
- configurations
 - activating 128
 - defined 11
 - displaying 31
 - displaying information 104, 105
 - general procedure 49
 - graph 154
 - viewing information 114
- configuring
 - applications 39, 41
 - disk groups 41
 - file systems 41
 - IP addresses 41
 - resources 41
- confirming
 - object actions 110
- context menus
 - clusterwide table 126
 - environment variables 110
 - exiting maintenance mode 152
 - from Cluster Admin 104
 - from RMS configuration tree 108
 - from RMS graph 163
 - RMS configuration tree 108
 - stopping RMS 135
 - switching applications 141
 - viewing application graph 158
 - viewing clusterwide table 123
 - viewing RMS graph 154
 - viewing subapplication graph 159
 - viewing switchlog 115
- controlled applications 17
 - failover 18
- ControlledShutdown attribute 227
- Controller objects 17, 18, 64, 217
 - dependencies 106
 - Follow mode 19
 - graph 161
 - resource wizard 41
 - Scalable mode 20
- controlling
 - RMS operation 128
- creating
 - application 68
 - second application 85
- D**
- Deact state 27
- deactivating
 - applications 143
- debug messages
 - severity level 120
- defining timeout 237
- DEMO turnkey wizard 41, 50, 71, 72
- dependencies
 - applications 106
- dependent resources 14
- Detach button, log viewer 117
- detaching
 - log view 117
- details
 - changing level of, in graphs 164
- detectors 13, 27
 - fault situations 197
 - illegal 187
 - RMS Wizard Kit 25
 - RMS Wizard Tools 25
 - starting 130
- DetectorStartScript attribute 227
- directed switch requests 212

- directories
 - RMS 36
- directory hierarchy
 - root directory 236
 - specifying root directory 236
- disk classes
 - as application resources 22
- displaying
 - see also* viewing
 - application states 113
 - current RMS configuration 31
 - environment variables 110, 112
 - HV_AUTOSTARTUP 138
 - HV_RCSTART 132
 - object states 113
 - state names,
 - clusterwide table 123
- dkconfig command 248
- dkmigrate command 248
- dkmirror command 248
- dktab command 248
- documentation
 - related 2
 - wizards 65
- double faults 204
 - and Halt attribute 204, 220
- E**
- echo service 48
- ELM *see* Enhanced Lock Manager
- Enhanced Lock Manager 12
 - enabling and disabling 234
- ENV object 34, 217
- environment variables 34
 - displaying 110, 112
 - HV_APPLICATION 176, 242
 - HV_APPLICATION_STATE_CHANGE_INFO, scalable controller script 175
 - HV_AUTORECOVER 242
 - HV_AUTORECOVER script
 - AutoRecover attempt flag 176
 - HV_AUTOSTARTUP 238
 - HV_AUTOSTARTUP_IGNORE 232
 - HV_AUTOSTARTUP_WAIT 233
 - HV_CHECKSUM_INTERVAL 233
 - HV_CONNECT_TIMEOUT 238
 - HV_FORCED_REQUEST 242
 - HV_FORCED_REQUEST script
 - forced request flag 176
 - HV_HOST_STATE_CHANGE_INFO, scalable controller script 175
 - HV_LAST_DET_REPORT 242
 - HV_LAST_DET_REPORT, script last detector report 176
 - HV_LOG_ACTION 238
 - HV_LOG_ACTION_THRESHOLD 235
 - HV_LOG_WARN_THRESHOLD 235
 - HV_MAXPROC 239
 - HV_MLOCKALL 239
 - HV_NODENAME 243
 - HV_NODENAME, script current object name 176
 - HV_OFFLINE_REASON 242
 - HV_OFFLINE_REASON, script reason for offline processing 176
 - HV_RCSTART 132, 239
 - HV_RCSTART, changing 132
 - HV_REALTIME_PRIORITY 240
 - HV_REQUESTING_CONTROLLER_R script forced request flag 175
 - HV_SCALABLE_CONTROLLER, scalable controller script 175
 - HV_SCALABLE_INFO, scalable controller script 175
 - HV_SCRIPT_TYPE 243
 - HV_SCRIPT_TYPE, script type 176
 - HV_SCRIPTS_DEBUG 240
 - HV_SYSLOG_USE 240

- HV_USE_ELM 234
- HV_WAIT_CONFIG 236
- hvenv and hvenv.local files 34
- NODE_SCRIPTS_TIME_OUT 243
- NODE_SCRIPTS_TIME_OUT, script timeout value for current object 176
- RELIANT_HOSTNAME 240
- RELIANT_INITSRIPT 241
- RELIANT_LOG_LIFE 236
- RELIANT_LOG_PATH 236
- RELIANT_PATH 236
- RELIANT_SHUT_MIN_WAIT 237
- RELIANT_STARTUP_PATH 241
- SCRIPTS_TIME_OUT 241
- ENVL object 34, 217
- errors
 - at initialization 198
 - during offline processing 201
 - in offline state 205
 - reaction to 202
- etc, RMS directory 36
- expert operations
 - disabling ELM 234
- F**
- failover 12
 - defined* 16
 - applications 16
 - controlled applications 18
 - nodes 16
 - RCFS file system 246
- fault script 203
- Faulted state 27, 144, 150
 - clearing 144, 208
 - FaultScript 29
- faults
 - clearing 144
 - display in clusterwide table 125
 - failover 74
 - maintenance mode 150
 - SysNode 210
- FaultScript attribute 29, 220
- features, market-specific 97
- file systems
 - as application resources 22, 39
 - Fsystem 41
 - resource type 25
 - site preparation 42
 - warning threshold 235
- Filter button 117
- fjsvwvbs 252
- fjsvwvcnf 252
- follow controllers 19
 - maintenance mode 148
- forced
 - online requests 209
 - operations, and status code 98
 - shutdown 134
- fsck_rcfs 246
- Fsystem, resource wizard 41
- G**
- Gds, resource wizard 41
- GENERIC turnkey wizard 41, 72, 86
- Global Disk Services 41
- global environment variables 34
- Global Link Services 41
- Gls, resource wizard 41
- graphs 11, 53, 154
 - affiliation names 166
 - appearance after shut down 167
 - application 158
 - changing detail level 164
 - composite subapplication 161
 - context menus 163
 - maintenance mode scope 148
 - reinitializing 32
 - resource names 166
 - subapplications 159
 - viewing application 158
 - viewing object details 157
- gResource objects 14, 64, 217
- GUI
 - configuration tree 104
 - input and message area 104
 - menu bar 103

- messages 104
- starting RMS 105
- H**
- Halt attribute 220
 - and double faults 204
- heartbeats 11, 13, 26
 - and SysNode faults 210
 - mode 234
 - timeout 130
 - UDP, and ELM 234
- high availability 1, 9
 - specifying applications 52
- HostName attribute 227
 - andOp objects 217
- hosts
 - site preparation 42
- HTML documentation 251
- HV_APPLICATION 176, 242
- HV_APPLICATION_STATE_CHANGE_INFO 175
- HV_AUTORECOVER 176, 242
- HV_AUTOSTARTUP 220, 238
 - changing 137
- HV_AUTOSTARTUP_IGNORE 232
- HV_AUTOSTARTUP_WAIT 233
 - and PartialCluster attribute 223
- HV_CHECKSUM_INTERVAL 233
- HV_CONNECT_TIMEOUT 238
- HV_FORCED_REQUEST 176, 242
- HV_HOST_STATE_CHANGE_INFO 175
- HV_LAST_DET_REPORT 176, 242
- HV_LOG_ACTION 238
- HV_LOG_ACTION_THRESHOLD 235
- HV_LOG_WARN_THRESHOLD 235
- HV_MAX_HVDISP_FILE_SIZE 239
- HV_MAXPROC 239
- HV_MLOCKALL 239
- HV_NODENAME 176, 243
- HV_OFFLINE_REASON 176, 242
- HV_RCSTART 239
 - changing 132
- HV_REALTIME_PRIORITY 240
- HV_REQUESTING_CONTROLLER 175
- HV_SCALABLE_CONTROLLER 175
- HV_SCALABLE_INFO 175
- HV_SCRIPT_TYPE 176, 243
- HV_SCRIPTS_DEBUG 240
- HV_SYSLOG_USE 240
- HV_USE_ELM 234
- HV_WAIT_CONFIG 236
- hvassert command 30
- hvatr command 30
- hvcn command 31, 130
 - h option 234
- hvconfig command 31
- hvconsoles 45
- hvdisp command 35, 112, 113, 232
 - a option 113
 - ENV and ENVL 112
 - file size 239
 - no display 228
- hvdist command 31
- hvdump command 31
- hvenv file 34
- hvenv.local file 35
- hvxec command 63
- hvgdmake command 31
- hvipalias 44
- hvmlogclean command 31
- hvrclv command 32, 132
- hvreset command 32
- hvsetenv command 32, 132, 138
 - HV_AUTOSTARTUP 238
 - HV_RCSTART 239
- hvshut command 32, 136, 137
 - defining timeout 237
- hvswh command 32, 140, 142
 - f option 209
- hvuutil command
 - a option 143
 - activating applications 143
 - c option 144, 209
 - clearing faults 144

Index

- clearing Wait state 145
- d option 143
- deactivating applications 143
- defined 33
- f option 143
- m and -M options 153
- maintenance mode 153
- o option 145
- s option 143
- Standby state 143
- stopping applications 143

hwv command 25

- defined 49
- operation mode 54
- resuming configuration 85

I

- I_List attribute 221
- IgnoreStandbyRequest attribute 228
- include, RMS directory 36
- Inconsistent state 28
- initialization 186
 - error during 198
 - objects 187
 - script, specifying 241
 - Unknown state 187
- InitScript 29
- inittab file *see* /etc/inittab
- intended state,
 - maintenance mode 28, 148
- interfaces, alternate 43
- IP addresses
 - defining resources 22
 - resource wizard 41
- lppaddress, resource wizard 41

J

- Java, trusted applets 101

K

- kernel,
 - and Enhanced Lock Manager 12
- killing a node 14, 17

L

- LAN interfaces 44
- LastDetectorReport attribute 228
- left pane
 - Cluster Admin 103, 104
- LEFTCLUSTER 210
- lib, RMS directory 36
- LieOffline attribute 228
- local environment variables 34
- locking process in memory,
 - base monitor 239
- locks, Enhanced Lock Manager 12
- log files
 - specify directory 236
 - switchlog 48, 211
 - system 48
 - time of preservation 236
 - viewing 114, 115
- log viewer
 - tabbed view, Cluster Admin 103
- logging in, Cluster Admin 99

M

- MA commands
 - clrcumonctl 247
 - clrcimonctl 247
- main menu
 - wizards 51
- maintenance mode 28, 146, 148
 - CLI operations 153
 - entering 146
 - exiting 149
 - fault clearing 150
 - follow controllers 148
 - forced exit 152
 - intended state 148
 - operating notes 148
 - setting 33
 - starting 146
 - status code when entering 98
 - stopping 149
- man page, for RMS commands 30
- management server 98

- manual pages
 - display 245
 - listing 245
- market-specific
 - applications 10
 - features 97
- MaxControllers attribute 221
- messages 186
 - debug 120
- MonitorOnly attribute 221
- mount_rcfs 246
- N**
- naming conventions, RMS 43, 69
- networks
 - site preparation 42
- NFS Lock Failover 46
- ngadmin 246
- node names
 - CF 97
- node names in configuration
 - files 42, 43
- NODE_SCRIPTS_TIME_OUT 176, 243
- nodes
 - defined* 11, 14
 - failover 16
 - killing 14, 17
 - objects 64
 - viewing attributes 114
- NoDisplay attribute 228
- non-basic settings, wizards 57
- NullDetector attribute 228
- O**
- object state timeouts,
 - tracking by base monitor 11
- object types
 - andOp 217
 - Controller 217
 - ENV 217
 - ENVL 217
 - gResource 217
 - orOp 218
 - SysNode 218
 - userApplication 218
- objects
 - activating applications 143
 - attributes 34
 - clearing a fault 144
 - clusterwide table 123
 - command pop-ups 163
 - confirming actions 110
 - context menus 126
 - Controller 17, 18, 64, 106, 161
 - dependencies 155
 - displaying states 113
 - environment variables 34
 - graph customization 164
 - gResource 14, 64
 - maintenance mode 149
 - relationships 155
 - resource types 33
 - RMS full graph 155
 - RMS tree 108
 - selecting 108
 - starting an application 140
 - switching applications 141
 - SysNode 14, 64, 108
 - taking application offline 142
 - userApplication 14, 64, 108, 137
 - viewing attributes 114
 - viewing details in graphs 157
- offline processing 14
 - definition 198
 - fault situations 201
 - requests 199
 - scripts 25
- Offline state 11, 27
 - OfflineDoneScript 29
 - OfflineScript 29
 - PostOfflineScript 29
- OfflineDoneScript attribute 29, 229
- OfflineFault state 28
- OfflineScript attribute 29, 221
- online processing 14
- online scripts 25
- Online state 11, 27

- OnlineScript 29
- PostOnlineScript 29
- PreCheckScript 29
- PreOnlineScript 29
- OnlinePriority attribute 221
- OnlineScript attribute 29, 222
- OnlineTimeout attribute 229
- operator
 - intervention 211
- operator privileges 99
- overriding
 - automatic application startup 137

P

- parallel application support 9
- PartialCluster attribute 222
 - and HV_AUTOSTARTUP_WAIT 233
- PAS commands
 - clmtest 247
 - mpcstat 247
- PCS
 - tabbed view, Cluster Admin 103
- PersistentFault attribute 223
- physical disks
 - state at initialization 197
- pop-up menu *see* context menu
- PostOfflineScript attribute 29, 223
- PostOnlineScript attribute 29, 223
- PreCheckScript attribute 29, 229
- PreOffline processing 199
- PreOfflineScript attribute 223
- PreOnlineScript attribute 29, 224
- PreserveState attribute 224
 - effect on fault processing 204
 - effect on fault processing
 - processing 203, 204
- primary management server 98
- PRIMECLUSTER 9
- priority switch
 - request 211
- PriorityList attribute 224
- privileges 99
- proactive scripts 13

R

- rcfs_fumount 246
- rcfs_list 246
- rcfs_switch 246
- rcsd 251
- rcsd.cfg 251
- Rcvm, resource wizard 41
- reactive scripts 13
- recovery timeout, UDP heartbeat 234
- recovery, heartbeat timeout 11
- related documentation 2
- Reliant Monitor Services
 - see also* RMS
 - components 26
 - high availability 11
 - overview 10
- RELIANT_HOSTNAME 74, 240
- RELIANT_INITSCRIPT 241
- RELIANT_LOG_LIFE 236
- RELIANT_LOG_PATH 37, 236
- RELIANT_PATH 35, 36, 236
- RELIANT_SHUT_MIN_WAIT 237
- RELIANT_STARTUP_PATH 241
- requests 186
 - offline 199
 - offline processing 199
- request-triggered scripts
 - InitScript 29
 - OfflineScript 29
 - OnlineScript 29
 - PreCheckScript 29
 - PreOnlineScript 29
- Resource attribute 229
 - Controller objects 217
- resource wizards
 - Cmdline 41
 - Controller 41
 - Fsystem 41
 - Gds 41
 - Gls 41
 - Ipaddress 41
 - Rcvm 41
 - Vxvm 41

- resources
 - clearing faulted 33
 - configuring 41
 - defining 22
 - executing scripts 54
 - file system entries 45, 47
 - LAN interfaces 44
 - monitoring 64
 - names in graphs 166
 - non-basic settings 58
 - object types 33
 - objects 64
 - scripts 27
 - shared remote entries 45
 - states 11
- right pane
 - Cluster Admin 104
 - log messages 116
- right-click, mouse
 - clusterwide table 126
 - fault clearing 151
- rKind attribute
 - gResource objects 217, 229
- RMS
 - clusterwide table 123
 - configuration tree 106
 - controlling operation 128
 - default directory 36
 - directory structure 36
 - full graph 154
 - graphs 53, 154
 - naming conventions 43, 69
 - node names 97
 - severity levels 120
 - shutdown, and graph
 - appearance 167
 - starting 128
 - startup,
 - and HV_AUTOSTARTUP 137
 - stopping 133
 - tabbed view, Cluster Admin 103
 - tree 106
- RMS base monitor 11
- RMS CLI
 - hvxexec 63
- RMS commands 30
 - see also* individual commands
 - administration procedures 97
 - hvasert 30, 249
 - hvatr 30
 - hvcn 31, 130, 249
 - hvconfig 31, 249
 - hvdisp 31, 112, 113, 249
 - hvdist 31, 249
 - hvdump 31, 250
 - hvenv.local 250
 - hvgdmake 31, 250
 - hvlogclean 31, 250
 - hvrclv 32, 132
 - hvreset 32, 250
 - hvsetenv 32, 132, 138, 250
 - hvshut 32, 136, 137, 250
 - hvswitch 32, 140, 142, 209, 250
 - hvthrottle 33, 250
 - hvutil 33, 143, 144, 145, 153, 209, 250
 - options 98
 - status codes 98
- RMS Troubleshooting Guide 115, 117
- RMS Wizard Kit 10, 23, 25
 - detectors 25
 - hwv command 25
 - scripts 25
- RMS Wizard Tools 10, 23
 - detectors 25
 - hwv command 25
 - resource types 25
 - scripts 25
- RMS wizards
 - configuring 23
- rName attribute
 - gResource objects 217, 230
- rolling upgrade 13
- root privileges 99
- running processes 22

- S**
- SA_blade.cfg 251
- SA_rccu.cfg 251
- SA_rps.cfg 251
- SA_rsb.cfg 252
- SA_scon.cfg 252
- SA_sspint.cfg 252
- SA_sunF.cfg 252
- SA_wtinps.cfg 252
- scalability 9, 20
- scalable controllers 20
- scon 251
- script top-of-tree 176, 242
- scripts 13, 29
 - Offline 25
 - offline processing 25
 - Online 25
 - proactive 13
 - reactive 13
 - resources 27
 - RMS Wizard Kit 25
 - timeout 241
- SCRIPTS_TIME_OUT 241
- ScriptTimeout attribute 224
- sdtool 251
- searching log text 122
- secondary management server 98
- secondary menus, wizards 56
- send clear-fault request 209
- severity levels
 - Alert 120
 - Critical 120
 - Debug 120
 - Emergency 120
 - Error 120
 - Info 120
 - Notice 120
 - Warning 120
- Shut 142
- shutdown
 - RMS 133
 - status code 98
- Shutdown Facility 17
 - and RMS 14
- ShutdownPriority attribute 225
- SIS
 - tabbed view, Cluster Admin 103
- SIS commands
 - dtopadmin 252
 - dtopd 252
 - dtopdbg 252
- site preparation 42
- software monitor
 - function 1
 - RMS 10
- split-brain condition
 - display in clusterwide table 125
- SplitRequest attribute 230
- Standby state 27
 - switching to 143
- StandbyCapable attribute 225
- starting
 - applications 140
 - applications, automatic 137
 - maintenance mode 146
 - RMS 128, 130
 - RMS, and heartbeat recovery timeout 13, 130
 - RMS, automatically 131
- starting RMS
 - system run level 132
- startup file, RMS 130
- state machine 11
- states 27
 - Deact 27
 - displaying information 123
 - Faulted 27, 144, 150
 - Inconsistent 28
 - Maintenance 28
 - nodes, detected by ELM 12, 131
 - nodes, detecting in RMS 11
 - of objects in the graph 11
 - Offline 27
 - OfflineFault 28
 - Online 27
 - Standby 27, 143
 - Unknown 28

- Wait 28, 145
 - Warning 28
 - state-triggered scripts
 - FaultScript 29
 - OfflineDoneScript 29
 - PostOfflineScript 29
 - PostOnlineScript 29
 - WarningScript 29
 - status codes, RMS CLI 98
 - status icons
 - faults, maintenance mode 150
 - maintenance mode 148
 - stopping
 - applications 142
 - maintenance mode 149
 - RMS 133, 136
 - subapplications 106
 - graph 159
 - sub-menus, wizards 56
 - summary table 123
 - switching
 - defined* 211
 - applications 32, 141
 - switchlog 211
 - panel 117
 - viewing 114, 115
 - switchlog file 48
 - SysNode 14
 - fault 210
 - initializing 187
 - object selection 108
 - object type 64, 218
 - switching application to 32
 - Wait state, clearing 145
 - system files,
 - and site preparation 42
 - system log 48
- T**
- tabbed view
 - Cluster Admin left pane 103
 - Cluster Admin, msg 104
 - Cluster Admin, rms&pcs 105
 - log viewer 116
 - tables
 - clusterwide 123
 - context menus 126
 - taking an application offline 142
 - timeouts
 - heartbeat recovery 11
 - Tools menu
 - Adaptive Services 154
 - trusted Java applets 101
 - turnkey wizards 41, 52, 65
 - DEMO 50, 71
 - GENERIC 86
 - ORACLE 41
 - R/3 41
 - See also* wizards 71
- U**
- UDP protocol
 - cluster heartbeat 11
 - RMS heartbeats 234
 - Unknown state 28
 - exiting 186
 - initial state 187
 - us, RMS directory 36
 - userApplication objects 14, 64, 187, 218
 - activating 143
 - clearing faults 144
 - hvswitch command 140
 - RMS tree 106
 - selection 108
 - state information 123
 - taking Offline 142
 - with hvshut 137
- V**
- variables, environment 34
 - vdisk 248
 - Verisign,
 - and trusted Java applets 101
 - view
 - Cluster admin 103
 - log viewer 116

Index

- log, attaching and detaching 117
 - message 104
 - viewing
 - see also* displaying
 - application logs 115
 - composite subapplication
 - graph 161
 - GUI message 104
 - object attributes 114
 - subapplication graphs 159
 - virtual representation 11
 - volume managers 1, 22
 - Vxvm, resource wizard 41
- W**
- Wait state 28, 145, 211
 - clearing faulted resources 33
 - clearing hung nodes 33
 - clearing SysNode 145
 - Warning state 28
 - WarningScript 29
 - WarningScript attribute 29, 226
 - Web-Based Admin View
 - login 99
 - primary management server 98
 - secondary management
 - server 98
 - Wizard Kit 23, 25
 - configuration 40
 - Wizard Tools 40
 - wizards
 - basic settings 57
 - configuring 40
 - DEMO turnkey 50
 - frequently used items 50
 - general description 40
 - GENERIC turnkey 86
 - hvxexec command 63
 - main menu 51
 - non-basic settings 57
 - ORACLE 41
 - R/3 41
 - resource wizards 41
 - secondary menus 56
 - sub-menus 56
 - turnkey 41, 52, 65
 - wizards *see* RMS wizards
 - wvCntl 252
 - wvGetparam 253
 - wvSetparam 253
 - wvstat 253

Fujitsu Siemens Computers GmbH
User Documentation
33094 Paderborn
Germany

Comments
Suggestions
Corrections

Fax: (++49) 700 / 372 00001

email: manuals@fujitsu-siemens.com
<http://manuals.fujitsu-siemens.com>

Submitted by

Comments on PRIMECLUSTER™
Reliant Monitor Services (RMS) with Wizard Tools (Solaris®, Linux®)
Configuration and Administration Guide



Fujitsu Siemens Computers GmbH
User Documentation
33094 Paderborn
Germany

Comments
Suggestions
Corrections

Fax: (++49) 700 / 372 00001

email: manuals@fujitsu-siemens.com
<http://manuals.fujitsu-siemens.com>

Submitted by

Comments on PRIMECLUSTER™
Reliant Monitor Services (RMS) with Wizard Tools (Solaris®, Linux®)
Configuration and Administration Guide



