PRIMECLUSTER™

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

Edition December 2003

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

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

Preface

Introduction

Using the Wizard Tools interface

Configuration example

Administration

Advanced RMS concepts

Troubleshooting

Non-fatal error messages

Fatal error messages

Console error messages

Continued ▶

Appendix—Operating system error numbers

Appendix—Object types

Appendix—Attributes

Appendix—Environment variables

Appendix—List of manual pages

Glossary

Abbreviations

Figures

Tables

Index

Contents

1 Preface . . . 1
1.1 About this manual . . . 1
1.2 Related documentation . . . 2
1.3 Conventions . . . 4
1.3.1 Notation . . . 4
1.3.1.1 Prompts . . . 4
1.3.1.2 Manual page section numbers . . . 5
1.3.1.3 The keyboard . . . 5
1.3.1.4 Typefaces . . . 5
1.3.1.5 Example 1 . . . 5
1.3.1.6 Example 2 . . . 6
1.3.2 Command syntax . . . 6
1.4 Important notes and cautions . . . 7

2 Introduction . . . 9
2.1 PRIMECLUSTER overview . . . 9
2.2 How RMS provides high availability . . . 10
2.2.1 Applications, resources, and objects . . . 10
2.2.2 Node and application failover . . . 11
2.2.3 Controlled applications and controller objects . . . 12
2.2.3.1 Follow controllers . . . 13
2.2.3.2 Scalable controllers . . . 14
2.2.3.3 Further notes about controllers . . . 15
2.3 How the Wizard Tools provide easy configuration . . . 16
2.4 How RMS Wizards provide easy configuration . . . 17
2.4.1 RMS Wizard Tools . . . 19
2.4.2 RMS Wizard Kit . . . 19
2.5 Cluster Admin . . . 20
2.6 RMS components . . . 20
2.6.1 Base monitor . . . 20
2.6.2 Detectors and states . . . 21
2.6.3 Scripts . . . 22
2.6.4 RMS CLI . . . 23
2.7 Object types . . . 26
2.8 Attributes . . . 26
2.9 Environment variables . . . 27
2.9.1 Setting environment variables . . . 28
2.10 Directory structure . . . 29

3 Using the Wizard Tools interface . . . 31
3.1 Overview . . . 31
3.1.1 RMS Wizard types . . . 32
3.1.1.1 Turnkey wizards . . . 32
3.1.1.2 Resource wizards . . . 33
3.2 Site preparation . . . 34
3.2.1 Network . . . 34
3.2.2 File systems—Solaris only . . . 36
3.2.2.1 NFS Lock Failover—Solaris only . . . 37
3.2.3 File systems—Linux only . . . 38
3.2.4 Log files . . . 39
3.2.5 Other system services and databases . . . 39
3.3 General configuration procedure . . . 40
3.4 Creating and editing a configuration . . . 40
3.4.1 Using the wizard menus . . . 41
3.4.2 Main configuration menu . . . 42
3.4.2.1 Main configuration menu when RMS is not active . . . 42
3.4.2.2 Main configuration menu when RMS is running . . . 46
3.4.3 Secondary menus . . . 46
3.4.4 Basic and non-basic settings . . . 47
3.5 Activating a configuration . . . 49
3.6 Configuration elements . . . 53
3.6.1 Scripts . . . 53
3.6.2 Detectors . . . 54
3.6.3 RMS objects . . . 54
3.7 Further reading . . . 55

4 Configuration example . . . 57
4.1 Stopping RMS . . . 57
4.2 Creating a configuration . . . 58
4.3 Adding hosts to the cluster . . . 58
4.4 Creating an application . . . 61
4.5 Entering Machines+Basics settings . . . 64
4.6 Entering non-basic settings . . . 68
4.7 Specifying a display . . . 70
4.8 Adding AlternateIps to the cluster (Linux only) . . . 73
4.9 Activating the configuration . . . 77
4.10 Creating a second application . . . 79
4.11 Setting up a controlling application . . . 83
4.12 Specifying controlled applications . . . 84
4.13 Activating the configuration a second time . . . 88
4.14 Starting RMS . . . 89

5 Administration . . . 91
5.1 Overview . . . 91
5.2 Using Cluster Admin . . . 91
5.2.1 Starting Cluster Admin . . . 91
5.2.2 Logging in . . . 93
5.2.3 Main screen . . . 95
5.2.4 RMS main window . . . 97
5.2.4.1 RMS tree . . . 97
5.2.4.2 Configuration information or object attributes . . . 99
5.2.4.3 Command pop-ups . . . 100
5.2.4.4 Confirmation pop-ups . . . 102
5.2.4.5 Switchlogs and application logs . . . 103
5.2.5 RMS graphs . . . 108
5.2.5.1 RMS full graph . . . 108
5.2.5.2 Application graph . . . 110
5.2.5.3 Subapplication graph . . . 111
5.2.5.4 Composite subapplication graph . . . 112
5.2.5.5 Configuration information from a graph . . . 113
5.2.5.6 Command pop-ups . . . 114
5.2.5.7 RMS graph customization . . . 115
5.2.5.8 Node status after RMS is shut down . . . 118
5.2.6 RMS clusterwide table . . . 119
5.2.6.1 Command pop-ups . . . 122
5.2.7 Changing the RMS configuration . . . 123
5.3 RMS procedures . . . 125
5.3.1 Starting RMS . . . 126
5.3.2 Stopping RMS . . . 130
5.3.3 Starting an application . . . 134
5.3.4 Switching an application . . . 136
5.3.5 Taking an application offline . . . 138
5.3.6 Activating an application . . . 138
5.3.7 Clearing a fault . . . 140
5.3.8 Clearing a sysnode Wait state . . . 140
5.3.9 Displaying environment variables . . . 142
5.3.10 Displaying application states . . . 145
5.3.11 Viewing the switchlog . . . 145
5.3.12 Viewing application logs . . . 146
5.3.13 Viewing GUI messages . . . 146

6 Advanced RMS concepts . . . 147
6.1 Internal organization . . . 147
6.1.1 Configuration structure . . . 147
6.1.2 Resource description . . . 148
6.1.3 Messages . . . 149

6.1.4 State transition rules . . . 149
6.2 States and scripts . . . 150
6.3 Initializing . . . 151
6.4 Online processing . . . 152
6.4.1 Online request . . . 152
6.4.1.1 Manual methods . . . 152
6.4.1.2 Automatic methods . . . 153
6.4.2 Online processing in a logical graph of a userApplication . . . 153
6.4.3 PreCheckScript . . . 155
6.4.4 Fault situations during online processing . . . 155
6.4.5 userApplication is already online . . . 156
6.5 Offline processing . . . 157
6.5.1 Offline request . . . 157
6.5.2 Offline processing in a logical graph of a userApplication . . . 157
6.5.3 Fault situations during offline processing . . . 158
6.5.4 Node is already offline . . . 159
6.5.5 Node does not have an Offline state . . . 159
6.6 Fault processing . . . 159
6.6.1 Faults in the online state or request processing . . . 159
6.6.2 Offline faults . . . 162
6.6.3 AutoRecover attribute . . . 162
6.6.4 Fault clearing . . . 163
6.6.5 SysNode faults . . . 164
6.6.5.1 Operator intervention . . . 165
6.7 Switch processing . . . 165
6.7.1 Switch request . . . 165
6.7.2 Extreme situations during switch processing . . . 167

7 Troubleshooting . . . 169
7.1 Overview . . . 169
7.2 Debug and error messages . . . 171
7.3 Log files . . . 172
7.4 Using the log viewer . . . 174
7.4.1 Search based on resource . . . 177
7.4.2 Search based on time . . . 178
7.4.3 Search based on keyword . . . 179
7.4.4 Search based on severity levels . . . 180
7.5 Using the hvdump command . . . 181
7.6 Specifying the log level . . . 182
7.7 Interpreting log files . . . 185
7.8 System log . . . 186
7.9 Wizard log files . . . 187
7.9.1 RMS Wizards detector logging . . . 189
7.9.2 Modifying levels while RMS is running . . . 190

7.10 PCS log files 191
7.10.1 Manual Script Execution 191
7.11 RMS troubleshooting 191

8 Non-fatal error messages 195
8.1 ADC: Admin configuration 196
8.2 ADM: Admin, command, and detector queues 205
8.3 BAS: Startup and configuration errors 227
8.4 BM: Base monitor 232
8.5 CML: Command line 243
8.6 CRT: Contracts and contract jobs 245
8.7 CTL: Controllers 246
8.8 CUP: userApplication contracts 247
8.9 DET: Detectors 248
8.10 GEN: Generic detector 252
8.11 INI: init script 253
8.12 MIS: Miscellaneous 254
8.13 NOD: Node detector 254
8.14 QUE: Message queues 259
8.15 SCR: Scripts 259
8.16 SWT: Switch requests (hvswitch command) 260
8.17 SYS: SysNode objects 262
8.18 UAP: userApplication objects 266
8.19 US: us files 270
8.20 WLT: Wait list 271
8.21 WRP: Wrappers 272

9 Fatal error messages 281
9.1 ADC: Admin configuration 282
9.2 ADM: Admin, command, and detector queues 282
9.3 BM: Base monitor 283
9.4 CML: Command line 285
9.5 CMM: Communication 285
9.6 CRT: Contracts and contract jobs 286
9.7 DET: Detectors 286
9.8 INI: init script 287
9.9 MIS: Miscellaneous 289
9.10 QUE: Message queues 289
9.11 SCR: Scripts 290
9.12 SYS: SysNode objects 292
9.13 UAP: userApplication objects 293
9.14 US: us files 293
9.15 WLT: Wait list 294
9.16 WRP: Wrappers 294

10 Console error messages 295
10.1 Console messages in alphabetical order 295

11 Appendix—Operating system error numbers 321

12 Appendix—Object types 323

13 Appendix—Attributes 325
13.1 Attributes available to the user 325
13.2 Attributes managed by configuration wizards 335

14 Appendix—Environment variables 341
14.1 Global environment variables 341
14.2 Local environment variables 345

15 Appendix—List of manual pages 349
15.1 CCBR 349
15.2 CF 349
15.3 CFS 350
15.4 CIP 350
15.5 Monitoring Agent 351
15.6 PAS 351
15.7 PCS 351
15.8 RCVM 352
15.9 Resource Database 352
15.10 RMS 353
15.11 RMS Wizards 354
15.12 SCON 355
15.13 SF 355
15.14 SIS 356
15.15 Web-Based Admin View 356

Glossary 357

Abbreviations 375

Figures 379

Tables 385

Index 387

1 Preface

PRIMECLUSTER™ Reliant® Monitor Services (RMS) is a software monitor designed to guarantee the high availability of applications in a cluster of nodes. After an introduction to RMS terminology and principles of operation, 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 and programmers familiar with installing and maintaining RMS configurations. Those who configure and administer RMS should be familiar with the following system functions and components:

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

1.1 About this manual

This manual is structured as follows:

- The chapter “Introduction” on page 9 contains general information on RMS and introduces the PRIMECLUSTER family of products.
- The chapter “Using the Wizard Tools interface” on page 31 describes how to configure RMS using the RMS Wizards.
- The chapter “Configuration example” on page 57 illustrates the Wizard configuration process for two simple applications on a small cluster.
- The chapter “Administration” on page 91 discusses how to administer RMS by means of the Cluster Admin GUI.
- The chapter “Advanced RMS concepts” on page 147 provides details about state detection and transition processing in RMS.
- The chapter “Troubleshooting” on page 169 describes how to troubleshoot RMS using graphical user interface (GUI) and command line interface (CLI) tools.
- The chapter “Non-fatal error messages” on page 195 lists all RMS error messages written to the log file along with their causes and resolutions.

- The chapter “Fatal error messages” on page 281 lists all fatal RMS error messages written to the log file along with their causes and resolutions.
- The chapter “Console error messages” on page 295 lists all RMS error messages written to the console along with their causes and resolutions.
- The chapter “Appendix—Operating system error numbers” on page 321 lists operating system error numbers for Solaris and Linux.
- The chapter “Appendix—Object types” on page 323 lists all the object types that are supplied with RMS.
- The chapter “Appendix—Attributes” on page 325 lists the attributes that are required for each object type.
- The chapter “Appendix—Environment variables” on page 341 describes the RMS environment variables.
- The chapter “Appendix—List of manual pages” on page 349 lists the manual pages for PRIMECLUSTER.

1.2 Related documentation

The documentation listed in this section contains information relevant to PRIMECLUSTER and can be ordered through your sales representative.

- *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) (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.
- *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 topics such as the configuration of file systems and IP addresses. They also describe the different kinds of wizards.

Suggested documentation

The following manuals contain information relevant to RMS administration and can be ordered through your sales representative (not available in all areas):

- *ANSI C Programmer's Guide*
- *LAN Console Installation, Operation and Maintenance*
- *Terminal TM100/TM10 Operating Manual*
- *PRIMEPOWER User's Manual* (operating manual)

Your sales representative will need your operating system release and product version to place your order.

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 *node#* indicates a root prompt on the specified node. For example, a command preceded by `fuji2#` would mean that the command was run as user `root` on the node named `fuji2`.

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.
Italic	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., <*nodename*>`RMS`; 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 (I).
I	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.

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 discusses the following:

- The section “PRIMECLUSTER overview” on page 9 describes how RMS functions within the PRIMECLUSTER family of products.
- The section “How RMS provides high availability” on page 10 describes how RMS supplies high availability.
- The section “How RMS Wizards provide easy configuration” on page 17 details the RMS Wizard products: RMS Wizard Tools and RMS Wizard Kit.
- The section “Cluster Admin” on page 20 introduces the Cluster Admin graphical user interface (GUI).
- The section “Object types” on page 26 introduces the RMS object types.
- The section “Attributes” on page 26 defines the RMS attributes.
- The section “Environment variables” on page 27 lists the RMS environment variables.
- The section “Directory structure” on page 29 lists and describes the RMS directory structure.

2.1 PRIMECLUSTER overview

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

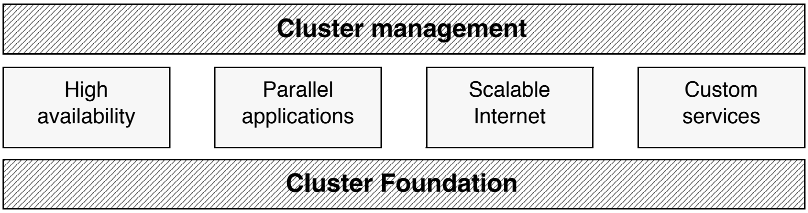


Figure 1: Overview of PRIMECLUSTER

The sections that follow focus on the role of the following PRIMECLUSTER products as they relate to high availability operation:

- 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 Wizard Tools and RMS Wizard Kit—These configuration products are used to create RMS configurations to control any number of user applications.
- Cluster Admin—The Cluster Admin GUI is the primary administrative tool for RMS.

RMS also provides integrated services for market-specific applications. See your sales representative for availability and details.

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

Within RMS, each resource used by an application is represented as an object, and each object is configured with the following:

- Detectors

- Scripts
- Dependent resources

RMS monitors each resource by using *detectors*, which are processes that report resource states to the RMS base monitor process. Resources are typically reported as *online* (enabled, available) or *offline* (disabled, unavailable), but a variety of other states is possible according to the type of resource.

Each resource type has an associated set of *scripts*. 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.

Internally, RMS represents a user application and all of its resources as a `userApplication` object. Bringing a `userApplication` to the online state, along with all of its dependant resources, is called *online processing*. Taking a `userApplication` to the offline state, along with all of its dependant resources, is called *offline processing*.

Machines that are members of a cluster are called *nodes*. Nodes that may run applications are represented by RMS `SysNode` objects. Like resource and application objects, each `SysNode` has an associated set of scripts and dependent resources.

2.2.2 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 `userAppli-cation` 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 `userAppli-`

cation 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.

2.2.3 Controlled applications and controller objects

In some situations, it is desirable for one application to control another in a parent/child relationship. Consider the scenario in Figure 2, in which a bank teller application depends on the network (represented by an Ipaddress subapplication) and a database application (which depends on a local file system represented by an Fsystem subapplication). If either the network or the database fails in some way, the parent teller application cannot complete any transactions. Therefore, from the RMS perspective, the database application acts as a resource that must be online if the teller application is to function properly.

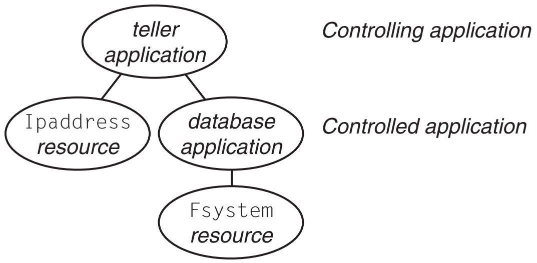


Figure 2: Controlled application scenario

RMS accommodates parent/child relationships between applications by providing a 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 3 demonstrates how RMS would represent the banking scenario. For the purposes of this example, only the application and controller objects are included in the illustration; resource objects representing network interfaces or file systems are not shown. Note that each controlled application requires a separate controller in the parent application, and that controllers exist only for internal RMS management purposes—there is no equivalent within the context of the user's applications.

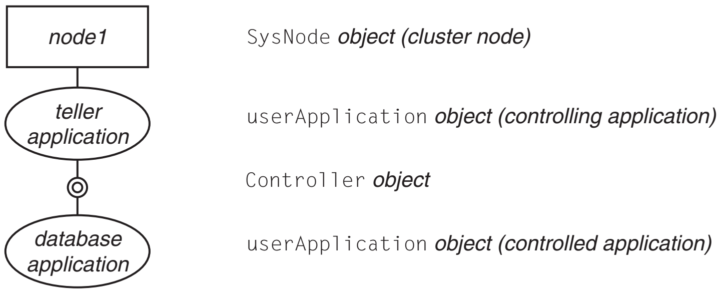

node1
SysNode object (cluster node)
teller application
userApplication object (controlling application)
Controller object
database application
userApplication object (controlled application)

Figure 3: RMS representation of controlled application

Failover of controlled applications

If a child changes to an offline or faulted state, RMS may switch the parent, the child, and all the dependent resources to other nodes. The exact action depends on whether the controller has been configured to operate in *Follow* or *Scalable* mode, as discussed below.

2.2.3.1 Follow controllers

When a controller operates in Follow mode, the corresponding child application must always run on the same node as the parent; that is, if the parent is switched to another node, the Follow-mode 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. This is illustrated in Figure 4.

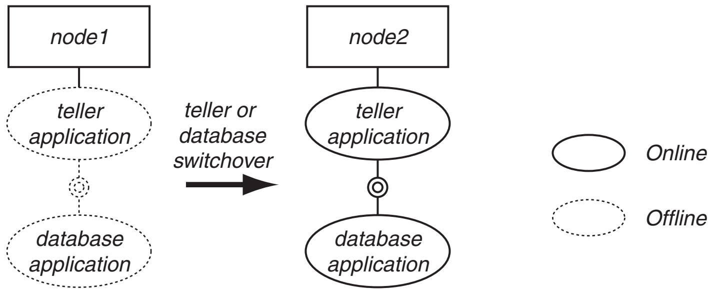


Figure 4: Follow mode switchover

Note the state of the Follow controller in Figure 4. 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 always run together on the same machine.

2.2.3.2 Scalable controllers

Scalable controllers allow the parent and child applications to run on separate machines. This not only allows more flexibility, but it may also prevent delays or outages when resources fail in certain combinations.

In the banking scenario, for example, the teller application depends on a network, and the database application depends on a local file system. 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, so the teller application is prevented from running on either `node1` or `node2`.

This will not happen if the controlled database application operates in Scalable mode. If the network is online on `node1`, and the file system is online on `node2`, then the database can be switched independently as shown in Figure 5.

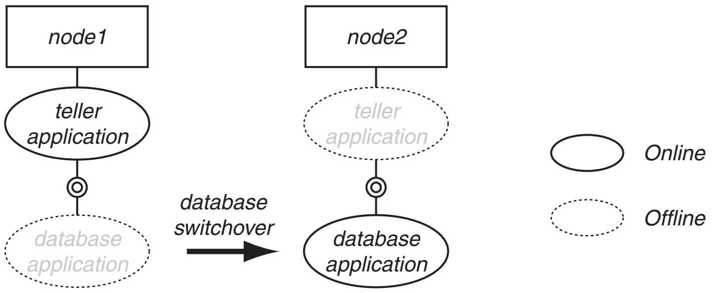


Figure 5: Scalable mode controlled (child) application switchover

Conversely, a network outage could cause RMS to switch the teller to `node2` while leaving the database online on `node1`, as shown in Figure 6.

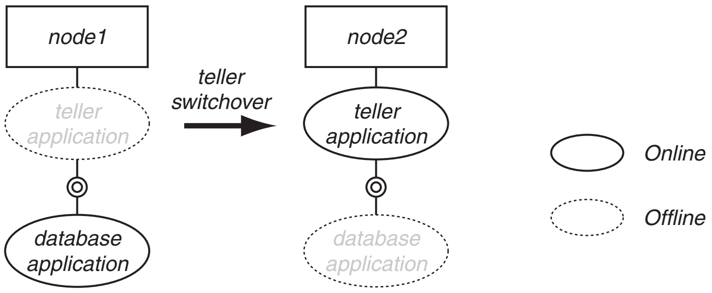


Figure 6: Scalable mode controlling (parent) application switchover

As noted earlier, RMS allows only one instance of an application in a 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 5 and Figure 6. For each Scalable-mode child application, an instance of its controller is online on every node where that application can run. This architecture allows RMS to efficiently monitor the cluster resources available to each child application, regardless of where the application is running at the time.

2.2.3.3 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.

However, a parent application can have more than one child application. Since each child has its own controller in the parent, each can operate in a different mode. For example, suppose the teller application in the banking scenario also has an ATM controlled application. The database could be configured to operate in Follow mode, while the ATM application could be configured to operate in Scalable mode.

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 dependant 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 How RMS Wizards provide easy configuration

PRIMECLUSTER provides the RMS Wizards to allow the creation of flexible and quality-tested RMS configurations. The RMS Wizards present a simple user interface that prompts you for details regarding the applications. The RMS 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 full RMS configurations that can control any other type of application.

Specialists skilled in popular applications and in RMS worked together to create the RMS Wizards. The RMS Wizards are broken up into the following separate products:

- RMS Wizard Tools
- RMS Wizard Kit

Figure 7 depicts the relationship between RMS, RMS Wizard Tools, and the RMS Wizard Kit.

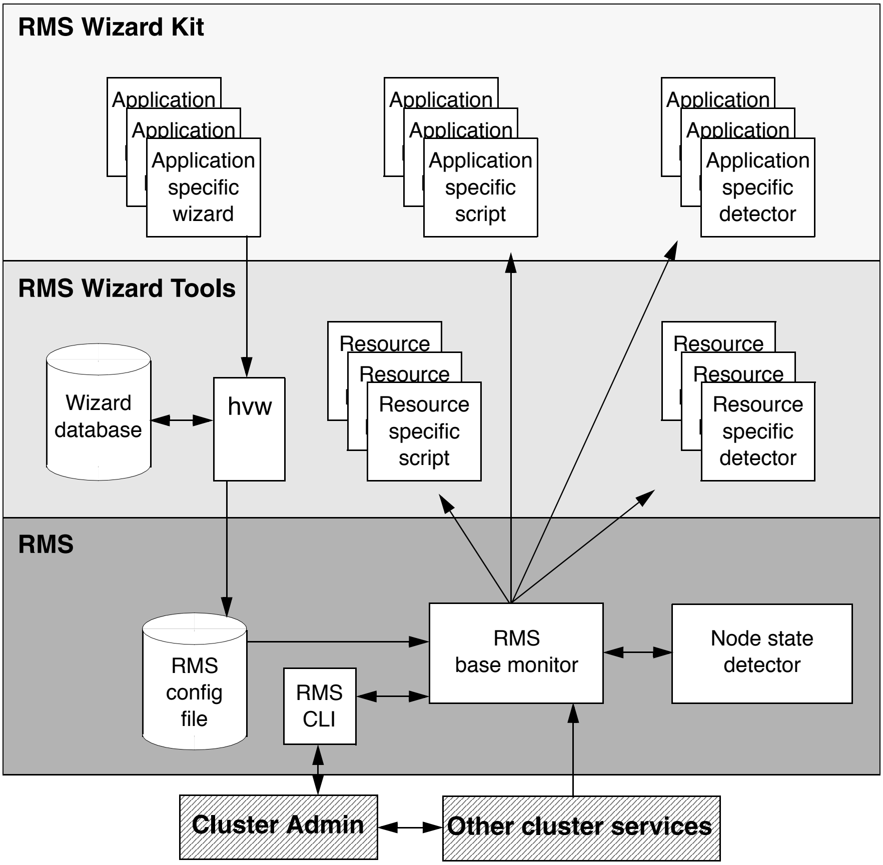


Figure 7: Relationship between RMS and RMS Wizards

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

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
- RMS CLI

2.6.1 Base monitor

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

- Stores the current configuration of resources as depicted by objects, their attributes, and their interdependent relationships
- Receives requests from the RMS command line interface (CLI) to take actions

- 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:

`Online`	Enabled, ready for use. All required children are online, and no errors were encountered while scripts were processed.
`Offline`	Disabled, not ready for use. The scripts have successfully deconfigured the resource.
`Faulted`	Error condition encountered. The error may have occurred in the resource, in one of its children, or during script processing.
`Standby`	Ready to be quickly brought `Online` when needed.
`Warning`	Some warning threshold has been exceeded. Also reported when: – a scalable controlled application is in transition from `Online` to `Offline`, or from `Standby` to `Faulted`; – a scalable controller object is `Online`, but some of its controlled applications are not; – a controlling application is `Online` but some of its scalable controller objects report `Warning`.
`OfflineFault`	Fault that occurred in the past has not yet been cleared.

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

`Wait`	Temporarily in transition to a known state. An action has been initiated for the affected resource, and the system is waiting for the action to be completed before allocating one of the above states.
`Unknown`	No information is available. Usually reported before object initialization is completed.
`Deact`	Applies to `userApplication` objects only. Operator intervention has deactivated the application throughout the cluster (such as for maintenance purposes).
`Inconsistent`	Applies to `userApplication` objects only. The object is `Offline` or `Faulted`, but one or more resource objects in its graph are `Online` or `Faulted`.

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 Tools and 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
  - `StateChangeScript`—Reaction to a scalable controller's `userAppli-cation` or `SysNode` changing 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 Wizards, and the primary interface for administering RMS is the Cluster Admin GUI. Both the RMS Wizards and Cluster Admin call the RMS CLI, and, under certain conditions, you may find the CLI useful. For example, to manually switch a user application to another node in the cluster, use the following CLI command:

► # **hvswitch** *userApplication SysNode*

In this case, *userApplication* is the user application that the user wants to switch to the system node *SysNode*.

Table 1 lists the RMS CLI commands available to administrators. Refer to the chapter “Appendix—List of manual pages” on page 349 for additional information on RMS CLI commands.

Command	**Function**
`hvassert`	Tests an RMS resource for a specified resource state. It can be used in scripts when a resource must achieve a specified state before the script can issue the next command. Does not require root privilege.
`hvattr`	Provides an RMS Wizard interface for changing the `AutoSwitchOver` attribute at runtime. The change can be made from a single node in the cluster and will be applied clusterwide for one or more `userApplication` objects in the currently running configuration. The values `No`, `HostFailure`, `ResourceFailure`, or `ShutDown` may be specified. `hvattr` command arguments are specific to RMS configuration files. The user should be familiar with the RMS Wizards.
`hvcm`	Starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the `hvcm` command. The base monitor is the decision-making module of RMS. It controls the configuration and access to all RMS resources. If a resource fails, the base monitor analyzes the failure and initiates the appropriate action according to the specifications for the resource in the configuration file.
`hvconfig`	Performs two tasks: displaying the current RMS configuration or sending the current configuration to an output file. The output of the `hvconfig` command is equivalent to the running RMS configuration file, but does not include any comments that are in the original file. Also, the order in which the resources are listed in the output might vary from the actual configuration file.
`hvdisp`	Displays information about the current configuration for RMS resources. Does not require root privilege.
`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 `.c` extension.

Table 1: Available CLI commands

Command	**Function**
`hvlogclean`	Saves old log files into a subdirectory whose name is the time RMS was last started (unless the `-d` option is used to delete the old log files instead). Regardless, `hvlogclean` creates a clean set of log files even while RMS is running.
`hvrclev`	Sets the RMS default-start run level to 3 to allow for the system processes started in the remote-file-sharing state as well as any user application resources started in run level 3. The `hvrclev` command can be used to reset the RMS default start run level back to the original run level 2. The `hvrclev` command is typically called from `pkgadd` to automatically adjust the RMS start run level for those customers who have a default system run level of 3.
`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 the entire graph will be brought back into a consistent initial state. **This command is intended for use by experts only.**
`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`.
`hvthrottle`	Prevents multiple scripts within a configuration file from running at the same time.
`hvutil`	Provides general administration interface to RMS. It performs various resource administration tasks, such as dynamically setting logging levels, sending a resource `Offline`, clearing faulted resources or hung cluster nodes in the `Wait` state, and setting detector time periods, 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 RMS Wizards, 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 323 for the supported types, their required attributes, and a description of each object.

i This information is provided for reference only. These objects are created by the RMS Wizards during the *Configuration—Activate* phase of the configuration process. Refer to the chapter “Using the Wizard Tools interface” on page 31.

2.8 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 325 for the supported types, their associated values, and a description of each attribute.

i This information is provided for reference material. The values are determined by the RMS Wizards during the *Configuration—Generate* phase of the configuration process. Refer to the chapter “Using the Wizard Tools interface” on page 31.

2.9 Environment variables

RMS uses global and local environment variables:

- Global variables must have the same setting on all nodes in the cluster. RMS maintains global environment variables in the `ENV` object and in the `/opt/SMAW/SMAWRrms/bin/hvenv` configuration file.
- Local variables override global variables and can differ from node to node. RMS maintains local environment variables in the `ENVL` object and in the `/opt/SMAW/SMAWRrms/bin/hvenv.local` configuration file.

If the `RELIANT_PATH` global variable has been redefined, global and local variables are located in the *RELIANT_PATH*`/bin/hvenv` and *RELIANT_PATH*`/bin/hvenv.local` files, respectively.

RMS creates the `ENV` or `ENVL` objects dynamically using the contents of the `hvenv` and `hvenv.local` files when the base monitor starts up. Values in the ENVL object override values in the ENV object. See the section “Setting environment variables” on page 28 for more details.

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.

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

The global environment variables (`ENV`) are as follows:

- `HV_AUTOSTARTUP_IGNORE`
- `HV_AUTOSTART_WAIT`
- `HV_CHECKSUM_INTERVAL`
- `HV_LOG_ACTION_THRESHOLD`
- `HV_LOG_WARN_THRESHOLD`
- `HV_WAIT_CONFIG`
- `RELIANT_LOG_LIFE`
- `RELIANT_LOG_PATH`
- `RELIANT_PATH`
- `RELIANT_SHUT_MIN_WAIT`

The local environment variables (`ENVL`) are as follows:

- `HV_CONNECT_TIMEOUT`
- `HV_LOG_ACTION`
- `HV_MAX_HVDISP_FILE_SIZE`
- `HV_MAXPROC`
- `HV_RCSTART`
- `HV_SYSLOG_USE`
- `RELIANT_HOSTNAME`
- `RELIANT_INITSCRIPT`
- `RELIANT_STARTUP_PATH`
- `SCRIPTS_TIME_OUT`

Refer to the chapter “Appendix—Environment variables” on page 341 for a description of all global and local environment variables.

2.9.1 Setting environment variables

When RMS starts, it reads the values of environment variables from `hvenv` and `hvenv.local` and initializes the `ENV` and `ENVL` objects respectively. To set the values of environment variables before starting RMS, the variables have to be specified in the `hvenv` and `hvenv.local` files.

A `/tmp` directory that is nearly full may result in RMS errors because `hvenv` uses this directory to sort RMS environment variables.

You can change the `hvenv.local` file on a node in the cluster, but the `hvenv` file must not be changed on any node. To activate your changes, you must stop RMS and restart it.

Caution

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

The values of environment variables are specified as export directives in these files. An example of an export directive would be as follows:

```
export SCRIPTS_TIME_OUT=200
```

You should change environment variables before running the configuration file. While RMS is running, you can display the environment variables with the `hvdisp` command, which does not require root privilege:

- `hvdisp ENV`
- `hvdisp ENVL`

2.10 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**
`RELIANT_PATH`	Base directory. **Default:** `/opt/SMAW/SMAWRrms`
<*RELIANT_PATH*>`/bin`	Executables, including detectors, commands, and scripts.
<*RELIANT_PATH*>`/build`	Work area for configuration files.
<*RELIANT_PATH*>`/etc`	Files that control the RMS environment.
<*RELIANT_PATH*>`/include`	RMS include files (header files) used by detectors and configuration files.
<*RELIANT_PATH*>`/lib`	RMS runtime libraries.
<*RELIANT_PATH*>`/us`	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`	Contains files that can be used for RMS analyzing and debugging. Detectors and `userApplication` objects create log files here when they are started. **Default:** `/var/opt/SMAWRrms/log`

Table 3: Log directory structure

3 Using the Wizard Tools interface

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

- The section “Overview” on page 31 gives a brief overall description of the configuration process and the RMS Wizards.
- The section “Site preparation” on page 34 describes the modifications to system files that are required for proper RMS operation.
- The section “General configuration procedure” on page 40 outlines the four major steps involved in every configuration procedure.
- The section “Creating and editing a configuration” on page 40 describes the wizard interface and how it is used to specify a configuration.
- The section “Activating a configuration” on page 49 describes how to activate a configuration after it has been created or modified.
- The section “Configuration elements” on page 53 provides additional details about basic RMS elements specified in every configuration.
- The section “Further reading” on page 55 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—These resource-oriented wizards provide scripts and detectors for basic resources such as file systems or IP addresses. The Wizard Tools also contain the GENERIC and DEMO application-oriented wizards.
- 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.

i 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 55 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 57 uses the DEMO and GENERIC 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).
- Ipaddress—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 (Solaris, Linux)* 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. See the section “Further reading” on page 55 for more details.

3.2.1 Network

- `/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	<*hostname*>`RMS`	`fuji2RMS` `fuji3RMS`
Alternate interfaces (`AlternateIps`)	<*hostname*>`rmsAI`<*nn*> where <*nn*> is a zero-filled sequence number in the range `01` to `99`	`fuji2rmsAI01` `fuji2rmsAI02`

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

i 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:

```
172.25.219.83    fuji2
172.25.219.84    fuji3
# host names for RMS
192.168.1.1      fuji2RMS
192.168.1.2      fuji3RMS
192.168.1.11     fuji2rmsAI01     # alt for fuji2
192.168.1.21     fuji2rmsAI02     # alt for fuji2
192.168.1.12     fuji3rmsAI01     # alt for fuji3
192.168.1.22     fuji3rmsAI02     # alt for fuji3
```

● `/.rhosts`

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 `fuji2` and `fuji3`, then every machine's `/.rhosts` file should contain the following lines:

```
fuji2 root
fuji3 root
```

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

  Contains entries for all of the LAN interfaces that are to be used as resources in the configuration.

  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 -n   IfName    Interface(s)   Netmask       Routes
fuji2    045dial   eth1           0xffffff00
```

- `/opt/SMAW/SMAWRrms/etc/hvconsoles`

  Controls customized handling of fault messages.

  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 fault information. A complete description of the format is available in the `hvcon-soles` 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. RMS entries appear as comments and will be ignored by all processes other than PRIMECLUSTER compo-nents. 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.

  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:fuji2:045nfs045dia1:045msg: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 8). 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 8: 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_SYS_LOG_USE = 0` in the `hvenv.local` file. The default is `1`.

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 (Solaris, Linux)*:

- `/etc/nsswitch.conf` system service lookup order database
- `rcp/rsh` service
- `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 130. 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 57 walks through an example.

► **Activate the configuration.**

Activation includes generation and distribution. See the section “Activating a configuration” on page 49.

► **Start RMS.**

Refer to the section “Starting RMS” on page 126. You can use the Cluster Admin GUI or the command line interface from any node in the cluster.

i To avoid network access problems, perform RMS configuration tasks as `root`, and ensure that `/.rhosts` and the `rcp`/`rsh` services are configured as described in the *Installation Guide*.

3.4 Creating and editing a configuration

You can bring up an existing wizard configuration that is running actively 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. 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 349 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* 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

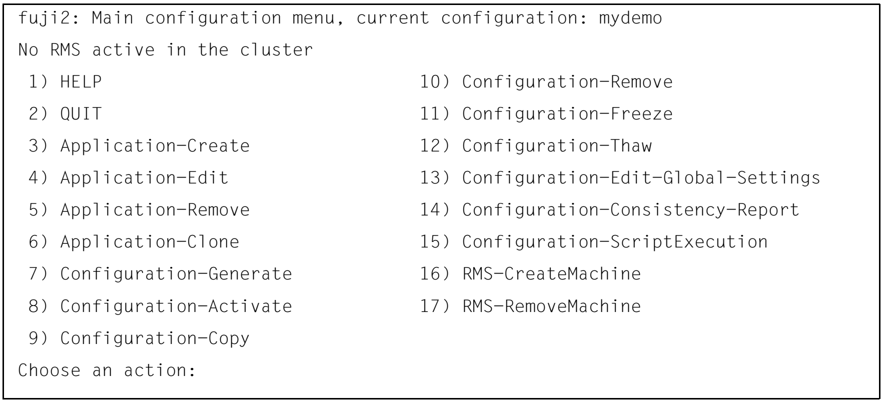
The *Main configuration menu* changes dynamically at run time depending on whether RMS is running in the cluster and whether the configuration being edited is the current configuration.

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 9 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 9: 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.

  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. Normally, you would use *Configuration-Activate* (described below) to generate and activate the configuration in one step.

  i *Configuration-Generate* is always available, whether RMS is running or not.
- *Configuration-Activate*—Generates and activates a configuration.

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

  The activation phase prepares the cluster for RMS, ensuring that all the required data is put into place. The wizard distributes the configuration data to every node and installs all necessary files.

  i *Configuration-Activate* is not available if RMS is already running on one or more nodes.
- *Configuration-Push*—Distributes a complete copy of the 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.

  i *Configuration-Push* is available only after the configuration has been activated.
- *Configuration-Copy*—Produces a copy of an existing configuration.

- *Configuration-Remove*—Removes an existing high-availability configuration.
- *Configuration-Freeze*—Prevents further changes to a configuration. With this option, the configuration can be viewed, but not modified.

  i *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 state.

  i *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 (AlternateIps) 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

When RMS is running on the local node, the *Main configuration menu* changes beginning with item 11, where the *Configuration-Push* menu item replaces the *Configuration-Activate* menu item (see Figure 10).

```
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 10: Main configuration menu when RMS is running

Configuration-Push provides the capability to update (push) the running configuration to another node that needs updating. For example, if one cluster node were down for maintenance, and you updated the RMS cluster configuration in the meantime, you could use *Configuration-Push* to update the node that was down for maintenance.

Item *12) RMS-ViewMachine* replaces the menu items that allow changes to the configuration when RMS is inactive.

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

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

Figure 11: Application type selection

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 57 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 12.

```
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 12: Menu leading to basic settings

If you select *7) Machines+Basics*, you can configure the basic settings using the menu in Figure 13. Items enclosed in parenthesis are optional.

```
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:
```

Figure 13: Menu to configure basic settings

After you complete the configuration of the basic settings, the non-basic settings menu appears (Figure 14). Non-basic settings include specifications for resources such as file systems, IP adresses, 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 14: Menu to configure non-basic settings

3.5 Activating a configuration

As described in section “General configuration procedure” on page 40, 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.

You must stop RMS before you activate a configuration.

The starting point for the activation phase is the *Main configuration menu* (see Figure 15).

```
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 15: 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 16).

```
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 16: Activating a configuration

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

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 nodes that are currently not running RMS will have the persistent status information removed during the *Configuration-Activate* process.

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 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: 2
```

Figure 17: 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:

```
hvcm -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* document, which is described in the section “Further reading” on page 55.

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.

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. The information is presented in separate directories covering the following major topics:

- *Primer*

  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 the *gresources* sub-section, which contains descriptions of a number of detectors. `Gresources` are defined as physical system resources.

- *Manual*

  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 manual pages.

Manual pages are available, for instance, for the `hvw` and the `hvexec` commands, which were also 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:

- “Creating a configuration” on page 58
- “Adding hosts to the cluster” on page 58
- “Creating an application” on page 61
- “Entering Machines+Basics settings” on page 64
- “Entering non-basic settings” on page 68
- “Specifying a display” on page 70
- “Adding AlternateIps to the cluster (Linux only)” on page 73
- “Activating the configuration” on page 77
- “Creating a second application” on page 79
- “Setting up a controlling application” on page 83
- “Specifying controlled applications” on page 84
- “Activating the configuration a second time” on page 88

An abbreviated version of this example appears in the *Installation Guide*.

i To avoid network access problems, perform RMS configuration tasks as `root`, and ensure that `/.rhosts` and the `rcp`/`rsh` services are configured as described in the *Installation Guide*.

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 130) 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/SMAW/SMAWRrms` 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 18).

```
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 18: 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 the section “Site preparation” on page 34).

i To override a default RMS primary host name, 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 add the host to the cluster in this step.

Select the nodes to be included in the configuration.The worksheet in Table 5 will be used as an aid to complete this configuration in an orderly fashion. See “Appendix—Cluster planning worksheet” in the *Installation Guide*.

Cluster Name:		FUJI		
	Cluster Console	RCA	Node 1	Node 2
Node Name	N/A	N/A	`fuji2`	`fuji3`
Public LAN				
Name	`fuji`	N/A	`fuji2`	`fuji3`
Device	`/dev/hme1`	N/A	`/dev/hme3`	`/dev/hme3`
IP Address	172.25.219.161	N/A	172.25.219.83	172.25.219.84
Netmask	255.255.255.0	N/A	255.255.255.0	255.255.255.0
Cluster Interconnect				
Device Name 1	N/A	N/A	`/dev/hme1`	`/dev/hme1`
Device Name 2	N/A	N/A	`/dev/hme2`	`/dev/hme2`
Device Name 3	N/A	N/A	`/dev/ip0`	`/dev/ip0`
Cluster IP				
Name	N/A	N/A	`fuji2RMS`	`fuji3RMS`
Address	N/A	N/A	192.168.1.1	192.168.1.2
Administrative LAN				
Name	`fujiSCON`	`fujiRCA`	`fuji2ADM`	`fuji3ADM`
Device	`/dev/hme0`	N/A	`/dev/hme0`	`/dev/hme0`
IP Address	172.25.200.1	172.25.200.2	172.25.200.4	172.25.200.5
Netmask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Table 5: Cluster site planning worksheet

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

```
# host names for RMS
192.168.1.1       fuji2RMS
192.168.1.2       fuji3RMS
192.168.1.11      fuji2rmsAI01      # alternate for fuji2
192.168.1.21      fuji2rmsAI02      # alternate for fuji2
192.168.1.12      fuji3rmsAI01      # alternate for fuji3
192.168.1.22      fuji3rmsAI02      # alternate for fuji3
```

In this step, you will add all of these hosts to the cluster.

► At the *Main configuration menu*, enter the number 16. The *Add hosts to a cluster* menu appears (Figure 19).

```
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 19: 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` `/etc/hosts` 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 19).

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 primary host names have been added, use *3) RETURN* to return to the *Main configuration menu*.

i By default, these host names are of the form *machinename*`RMS` to follow the RMS naming convention. To override the default RMS name for a machine, modify that machine's `hvenv.local` file and set the `RELIANT_HOSTNAME` variable to the desired name. This must be done before you add the machine to the cluster in this step.

To remove a node, select *17) RMS-RemoveMachine* from the *Main configuration menu*. The *Remove hosts from a cluster* menu appears (Figure 20).

```
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 20: 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 21).

```
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 21: Main configuration menu

► Select *Application-Create* by entering the number 3. The *Application type selection menu* appears (Figure 22).

```
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 22: Application type selection menu

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 79.

► 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 23).

```
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 23: 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 24).

```
onsistency 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 24: 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 25).

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

Figure 25: List of nodes for failover procedure

i The Wizards retrieve the default list of nodes from the CIP configuration file.

Since our application is presently configured for `fuji2RMS`, `fuji3RMS` should become the additional node:

► Select *fuji3RMS* by entering the number 4.

In the menu that follows (Figure 26) 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 (app1: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 26: 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 27).

```
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 27: 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 28: Setting flags for AutoSwitchOver mode

► Enter the number 7 for *RESOURCEFAILURE* (see Figure 28). 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 28).

You will be returned to the *Machines+Basics* menu (Figure 29). Note that item 16 now displays the *AutoSwitchOver* flags you just set.

```
Consistency check ...



Machines+Basics (app1: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 29: 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 30).

```
YConsistency 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 30: Non-basic settings

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 31).

```
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 31: 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.

Items in the menu body indicate which scripts the wizard provides for starting, stopping, and checking: see the lines beginning with *6) StartCommands[0]=*, *7) StopCommands[0]=*, and *8) CheckCommands[0]=*.

i 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 32).

```
 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 32: List of display options

You can choose from the list of detected 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 33).

```
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 33: 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 33), select *SAVE+EXIT* by entering the number 3.

This will take you back to the *Settings of turnkey wizard "DEMO"* menu (Figure 34).

```
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 34: 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 Adding AlternateIps to the cluster (Linux only)

To maintain high availability, RMS can employ multiple physical network connections to each host in the cluster. For RMS purposes, one connection to each machine is associated with the primary host name. Redundant connections to the same machine are associated with alternate interfaces known as `AlternateIps`. For high-reliability operation, `AlternateIps` should be included in the configuration.

In our example, both `fuji2` and `fuji3` have a total of three connections to the network. (See the `/etc/hosts` entries in the section “Adding hosts to the cluster” on page 58.) The primary host names were specified when the cluster was defined. In this step, two `AlternateIps` will be added for each machine.

Configure your applications and all their associated nodes (`Machines[]` lists) before you add `AlternateIps`. If a node is not used by any application, neither its primary name nor its `AlternateIps` will be available in the menus described below.

► From the *Main configuration menu*, select *15) Configuration-Edit-Global-Settings*. The *Global settings: main menu* appears (Figure 35).

```
Global settings: main menu (consistent):
 1) HELP                                 7) MaxAlternateIps=
 2) NO-SAVE+EXIT                         8) PreCheckTimeout=
 3) SAVE+EXIT                            9) FirstAvailableDetector=0
 4) ShowTurnkeyWizardsOnly              10) LastAvailableDetector=127
 5) AdditionalAlternateIps              11) MaxMenuItemsDisplayed=
 6) AdditionalI_List                    12) DetectorDetails
Choose the global setting to process:
```

Figure 35: Global settings: main menu

► Select *5) AdditionalAlternateIps*. The *Global settings: machines menu* appears (Figure 36).

```
Global settings: machines menu
1) HELP
2) RETURN
3) MORECHOICES
4) fuji2RMS
5) fuji3RMS
Choose a host which needs additional RMS AlternateIps:
```

Figure 36: Global settings: machines menu

Starting with item 4, this menu lists all cluster hosts that are already used by at least one application. The menu does not show hosts that are unused.

- ► Select *4) fuji2RMS*. The *Global settings: AlternateIps* first menu for `fuji2RMS` appears (Figure 37).

```
Global settings: AlternateIps for fuji2RMS
1) HELP                                4) NONE
2) NO-SAVE                             5) AdditionalAlternateIps
3) SAVE
Choose the RMS IpAlias to process:
```

Figure 37: Global settings: AlternateIps first menu

- ► Select *5) AdditionalAlternateIps*. The *Global settings: AlternateIps* second menu for `fuji2RMS` appears (Figure 38).

```
Global settings: AlternateIps for fuji2RMS
1) HELP                                5) fuji2rmsAI02
2) RETURN
3) FREECHOICE
4) fuji2rmsAI01
Choose the RMS IpAlias:
```

Figure 38: Global settings: AlternateIps second menu

- ► Select *4) fuji2rmsAI01*. The *Global settings: AlternateIps* first menu for `fuji2RMS` appears (Figure 39).

```
Global settings: AlternateIps for fuji2RMS
1) HELP                                 4) NONE
2) NO-SAVE                              5) AdditionalAlternateIps
3) SAVE                                 6) IpAliasForM[0]=fuji2rmsAI01
Choose the RMS IpAlias to process:
```

Figure 39: Global settings: AlternateIps first menu with first interface

Repeat the previous two steps, but this time choose *5) fuji2rmsAI02*. The *Global settings: AlternateIps* first menu for `fuji2RMS` will then appear with both `AlternateIps` (Figure 40).

```
Global settings: AlternateIps for fuji2RMS
1) HELP                                 5) AdditionalAlternateIps
2) NO-SAVE                              6) IpAliasForM[0]=fuji2rmsAI01
3) SAVE                                 7) IpAliasForM[1]=fuji2rmsAI02
4) NONE
Choose the RMS IpAlias to process:
```

Figure 40: Global settings: AlternateIps first menu with both interfaces

► Select *3) SAVE*. This will save the list of `AlternateIps` for `fuji2RMS` and return you to the *Global settings: main menu*, which has been updated with the new information (Figure 39).

```
Global settings: main menu (consistent):
 1) HELP
 2) NO-SAVE+EXIT
 3) SAVE+EXIT
 4) ShowTurnkeyWizardsOnly
 5) AdditionalAlternateIps
 6) AdditionalI_List
 7) IpAliases[0]=fuji2RMS/fuji2rmsAI01,fuji2rmsAI02
 8) MaxAlternateIps=
 9) PreCheckTimeout=
10) FirstAvailableDetector=0
11) LastAvailableDetector=127
12) MaxMenuItemsDisplayed=
13) DetectorDetails
Choose the global setting to process:
```

Figure 41: Global settings: main menu with AlternateIps for first host

Item *7) IpAliases[0]* now displays `fuji2RMS` and the names that correspond to its alternate interfaces. Note that the menu header now indicates the configuration is not yet consistent, and the reason for the status change: `fuji3RMS` has `AlternateIps` that have not yet been added to the cluster.

Repeat the above process for `fuji3RMS`, this time adding `fuji3rmsAI01` and `fuji3rmsAI02` to the cluster. The final *Global settings: main menu* should appear as shown in Figure 39.

```
Global settings: main menu (consistent):
 1) HELP
 2) NO-SAVE+EXIT
 3) SAVE+EXIT
 4) ShowTurnkeyWizardsOnly
 5) AdditionalAlternateIps
 6) AdditionalI_List
 7) IpAliases[0]=fuji2RMS/fuji2rmsAI01,fuji2rmsAI02
 8) IpAliases[1]=fuji3RMS/fuji3rmsAI01,fuji3rmsAI02
 9) MaxAlternateIps=
10) PreCheckTimeout=
11) FirstAvailableDetector=0
12) LastAvailableDetector=127
13) MaxMenuItemsDisplayed=
14) DetectorDetails
Choose the global setting to process:
```

Figure 42: Global settings: main menu with AlternateIps for both hosts

Select *3) SAVE+EXIT* to save the updated information and return to the *Main configuration menu*.

4.9 Activating the configuration

As described in the section “General configuration procedure” on page 40, 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 43).

```
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 43: 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 49. You will be prompted to continue at the end of the process (see Figure 44).

```
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 44: Successful configuration activation

► Press the [Enter] or [Return] key to return to the *Main configuration menu* (Figure 45).

```
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 45: 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.

4.10 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 46).

```
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 46: 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 47).

```
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 47: 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 48).

```
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 48: 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 49).

```
Consistency check ...

Machines+Basics (app2: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 49: Machines+Basics menu

► Select *AdditionalMachine* by entering the number 5. A menu appears with the list of available machines (Figure 50).

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

Figure 50: 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 (Figure 51). The *8) Machines[1]* item now displays `fuji3RMS` as the additional machine. `APP2` will be switched over to this machine if `fuji2RMS` fails.

```
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 51: Machines+Basics menu

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.

4.11 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 52).

```
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 52: 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 55).

```
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 53: Assigning a controller

4.12 Specifying controlled applications

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

► Select *AdditionalAppToControl* by entering the number `6`.

The menu that appears offers you a list from which to choose an application (Figure 54).

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

Figure 54: 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 55).

```
Set flags for (sub) application: app1
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 55: 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 56), select *FREECHOICE* by entering the number `3`.

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

Figure 56: 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 57).

```
Set flags for (sub) application: app1
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 57: 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 58).

```
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:app1
8) (InParallel=)
9) (FaultScript=)
Choose the setting to process: 3
```

Figure 58: 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 59).

```
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(Ctl_APP2)             17) Gds:Global-Disk-Services(-)
 9) LocalFileSystems(-)               18) Gls:Global-Link-Services(-)
Choose the setting to process: 3
```

Figure 59: 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 60).

```
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 60: Main configuration menu

This completes the creation of the second application.

4.13 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 61).

```
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 61: 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 49. You will be prompted to continue at the end of the process (see Figure 54).

```
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 62: Activating the configuration for the second time

► Press the [Enter] or [Return] key to return to the *Main configuration menu* (Figure 63).

```
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 63: Return to Main configuration menu

► Select *QUIT* by entering the number 2.

This ends the activation phase of the configuration process.

4.14 Starting RMS

At this point, you are ready to start RMS to monitor both applications.You can use the Cluster Admin GUI (see the section “Starting RMS” on page 126) or you can enter the following command from any machine in the cluster:

```
# hvcm -a mydemo
```

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 discusses the following:

- The section “Overview” on page 91 introduces PRIMECLUSTER administration by means of the Cluster Admin and the CLI.
- The section “Using Cluster Admin” on page 91 discusses how to use the RMS portion of the GUI.
- The section “RMS procedures” on page 125 describes how to Administer RMS using the GUI. It also contains CLI commands as a convenience for advanced users.

5.1 Overview

RMS administration can be done by means of the Cluster Admin GUI or by the CLI; however, it is recommended that you use the Cluster Admin GUI. The CLI should only be used by expert system administrators or in cases where a browser is not available. The following sections primarily describe the Cluster Admin GUI options. The CLI equivalents are provided in the RMS procedures section.

5.2 Using Cluster Admin

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

i Windows desktop systems require the Java plug-in as specified in the *Web-Based Admin View Operation Guide*.

5.2.1 Starting Cluster Admin

Open the Java-enabled browser (use Internet Explorer 5.*x* or Netscape Navigator 4.*x* or higher versions) 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`

After contacting the host, the browser changes the URL suffix from `.cgi` to `.html`. Figure 64 shows an example of the Cluster Admin opening screen. For details on the primary and secondary management servers, please refer to the *Web-Based Admin View Operation Guide*.

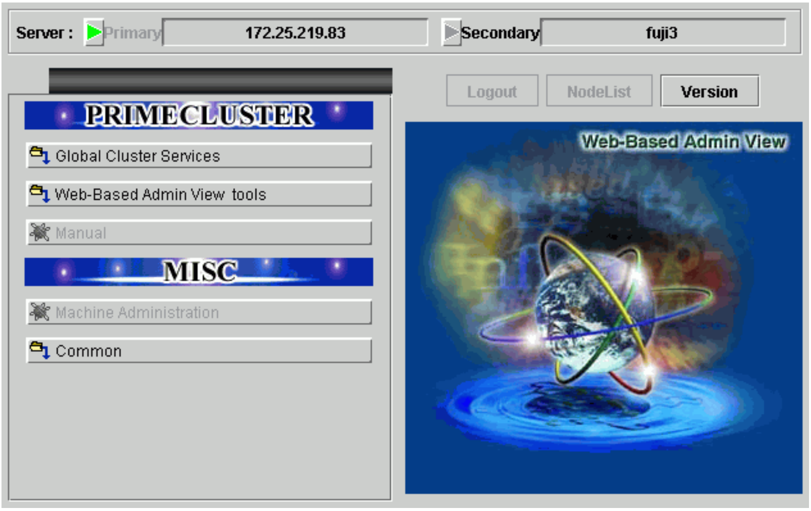


Figure 64: Invoking the Cluster Admin GUI

5.2.2 Logging in

After the Web-Based Admin View login screen appears (Figure 65), log in as follows:

► Enter the user name and password for a user with the appropriate privilege level.

► Click on the *OK* button.

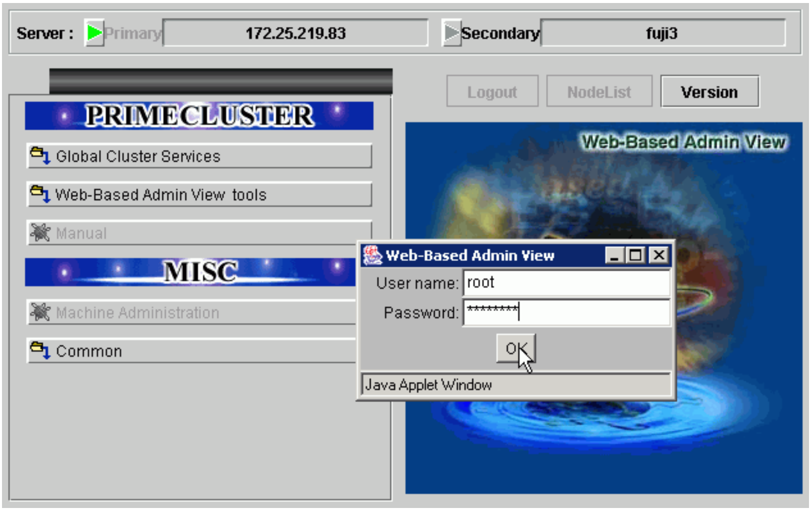


Figure 65: Web-Based Admin View login screen

Use the appropriate privilege level while logging in. 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 the privilege levels, refer to the *PRIMECLUSTER Installation Guide (Solaris, Linux)*.

After clicking on the *OK* button, the top menu appears (Figure 66).

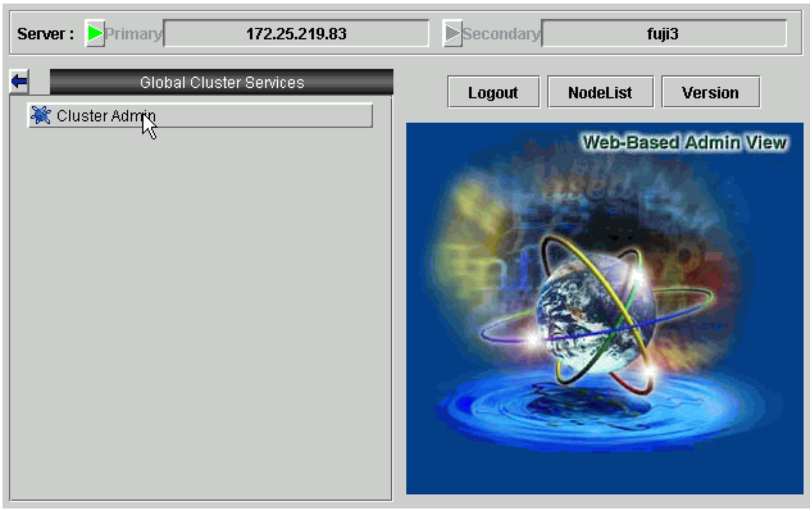
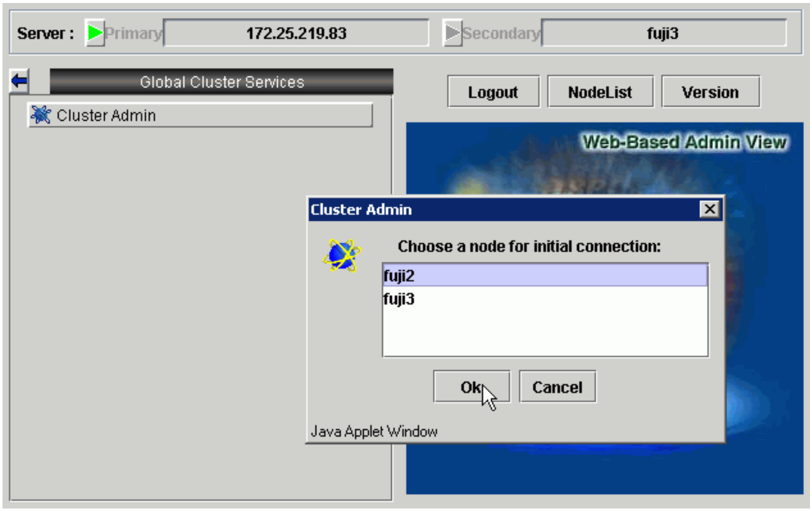


Figure 66: Top menu

Open Cluster Admin as follows:

1. Click on *Global Cluster Services*.
2. Click on the *Cluster Admin* button to start Cluster Admin
3. The *Choose a node for initial connection* screen appears (Figure 67). Select a node, and click on *OK*. The main Cluster Admin screen appears.



Choose a node for initial connection:

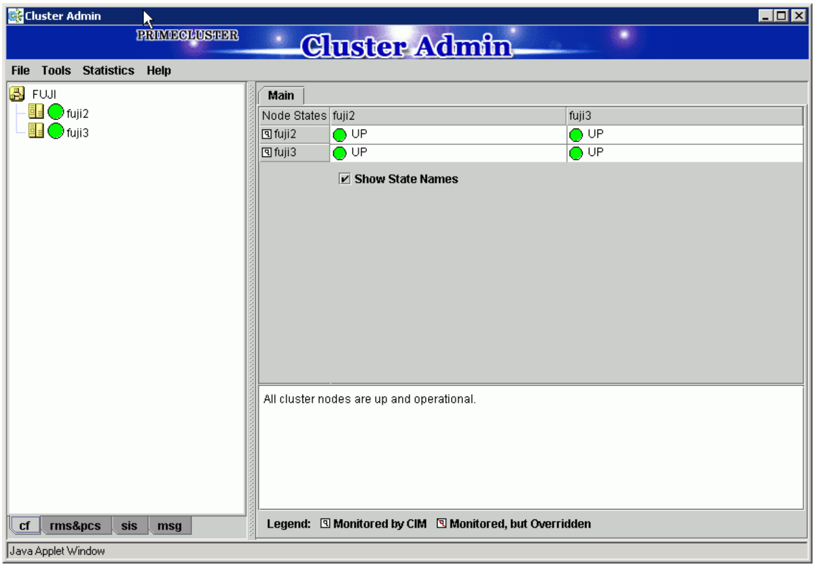
fuji2
fuji3

Figure 67: Cluster menu

5.2.3 Main screen

The main screen (Figure 68) contains the following tabs on the left-hand side panel:

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



Node States	fuji2	fuji3
fuji2	● UP	● UP
fuji3	● UP	● UP

☑ Show State Names

All cluster nodes are up and operational.

Legend: Monitored by CIM Monitored, but Overridden

Figure 68: Main screen

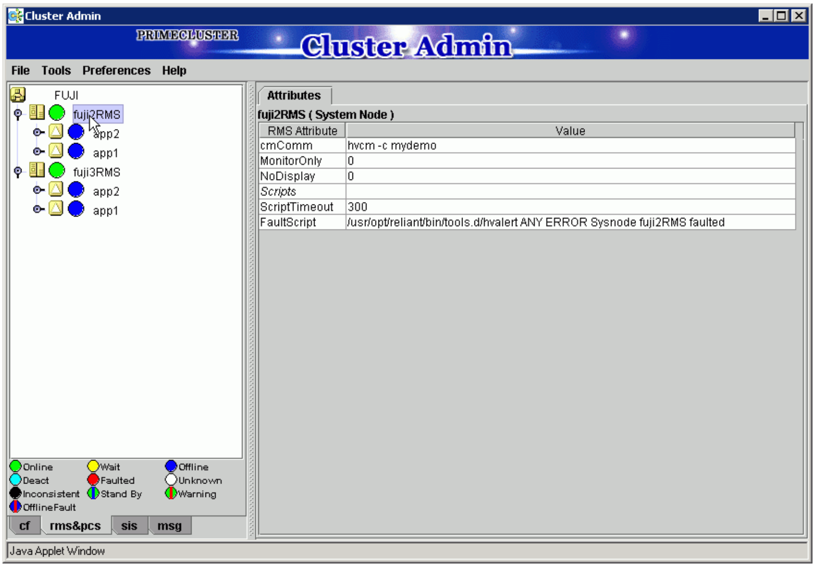
Select the appropriate tab to switch to a component. By default, the *cf* tab is selected.

The Cluster Admin GUI has some standard components that are common across RMS, CF, SIS, and the message window. They are as follows:

- Pull-down menus—Pull-down menus that contain both functions generic to the Admin GUI and specific to the PRIMECLUSTER products.
- Tree panel—Panel on the left is normally the tree panel. This panel displays product-specific configuration information. Click on a tree component to view further information in the main panel.
- Main panel—Large panel 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.

5.2.4 RMS main window

To start the RMS portion of the GUI, click on the *rms* tab. An example of the RMS main window is shown in Figure 69. The main window area is split into two sub-areas. The RMS tree is displayed on the left-hand side panel. The right-hand side panel is used to display configuration information or properties of nodes, logs, or both, depending on the selections in the RMS tree.



fuji2RMS (System Node)

RMS Attribute	Value
cmComm	hvcm -c mydemo
MonitorOnly	0
NoDisplay	0
Scripts	
ScriptTimeout	300
FaultScript	/usr/opt/reliant/bin/tools.d/hvalert ANY ERROR Sysnode fuji2RMS faulted

Figure 69: RMS main window

5.2.4.1 RMS tree

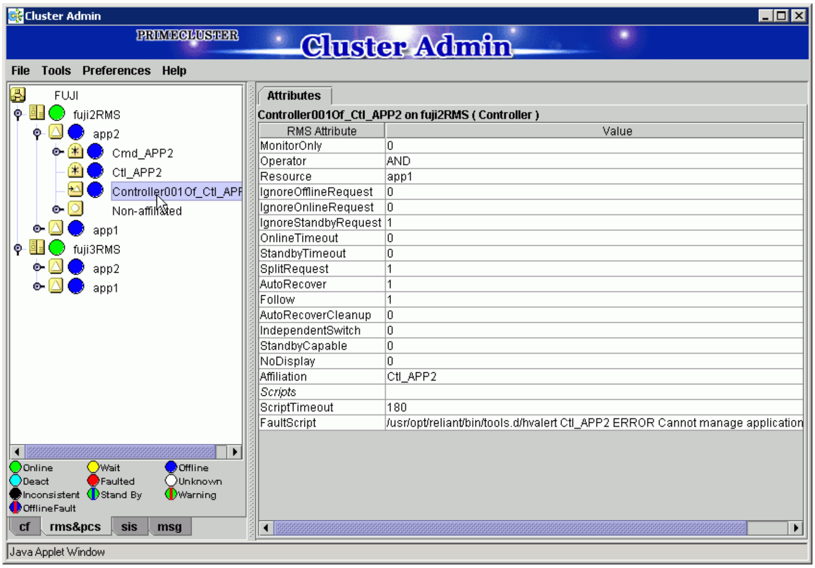
The RMS tree displays the configuration information of the cluster in a hierarchical format. The tree has the following levels:

- Root of the tree—Represents the cluster.
- First level—Represent the system nodes forming the cluster.
- Second level—Represent the `userApplication` objects running on each of the system nodes.
- Third level—Represent subapplications, if any.

- Fourth level—Represents the resources necessary for each of the 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`.

Dependencies between the applications are depicted in the RMS tree by means of the controller object. An example of the RMS tree with a controller object is shown in Figure 70.



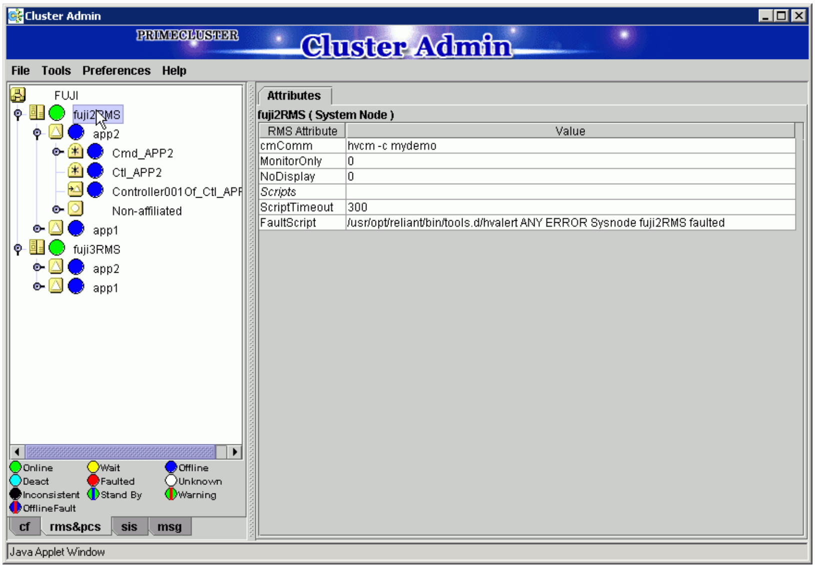
Controller001Of_Ctl_APP2 on fuji2RMS (Controller)

RMS Attribute	Value
MonitorOnly	0
Operator	AND
Resource	app1
IgnoreOfflineRequest	0
IgnoreOnlineRequest	0
IgnoreStandbyRequest	1
OnlineTimeout	0
StandbyTimeout	0
SplitRequest	1
AutoRecover	1
Follow	1
AutoRecoverCleanup	0
IndependentSwitch	0
StandbyCapable	0
NoDisplay	0
Affiliation	Ctl_APP2
Scripts	
ScriptTimeout	180
FaultScript	/usr/opt/reliant/bin/tools.d/hvalert Ctl_APP2 ERROR Cannot manage application

Figure 70: RMS tree with a controller object

5.2.4.2 Configuration information or object attributes

View the configuration information for the 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-hand side panel of the RMS main window (Figure 71).



fuji2RMS (System Node)

RMS Attribute	Value
cmComm	hvcm -c mydemo
MonitorOnly	0
NoDisplay	0
Scripts	
ScriptTimeout	300
FaultScript	/usr/opt/reliant/bin/tools.d/hvalert ANY ERROR Sysnode fuji2RMS faulted

Figure 71: Configuration information or object attributes

5.2.4.3 Command pop-ups

You can perform many operations on the RMS tree objects by using the context-sensitive command pop-up menus. Invoke the pop-up menu by right-clicking with the mouse on the object. The menu options are based on the type and the current state of the selected object (Figure 72).

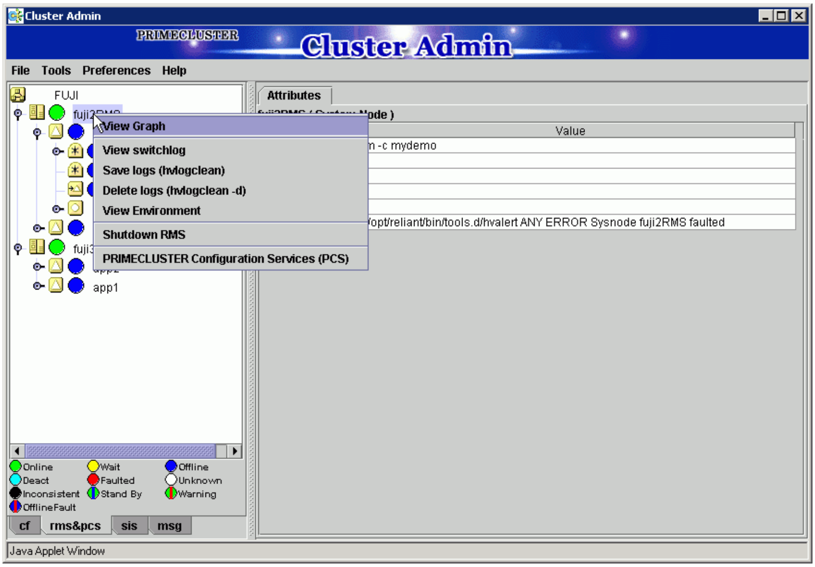
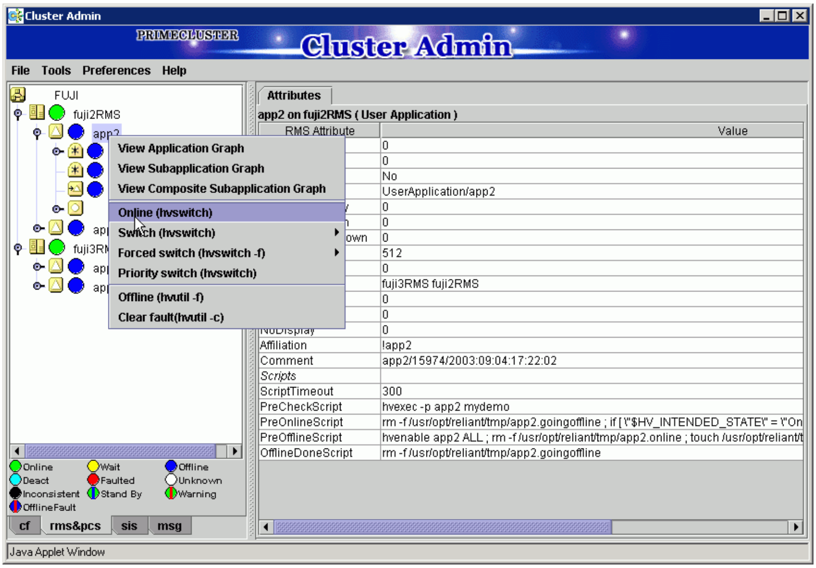


Figure 72: Command pop-up

For example, the menu offers different options for a `SysNode` object selection and `userApplication` object selection. It also offers different options for a `userApplication` object in the online state than in the offline state (Figure 73).



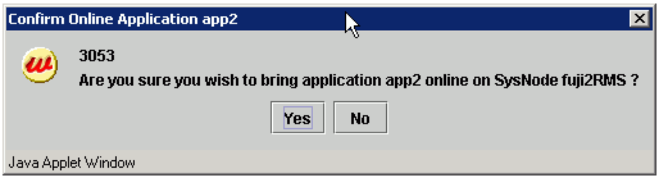
app2 on fuji2RMS (User Application)

RMS Attribute	Value
	0
	0
	No
	UserApplication/app2
y	0
n	0
own	0
	512
	0
	fuji3RMS fuji2RMS
	0
	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"On
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/t
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 73: Command pop-up for an offline application

5.2.4.4 Confirmation pop-ups

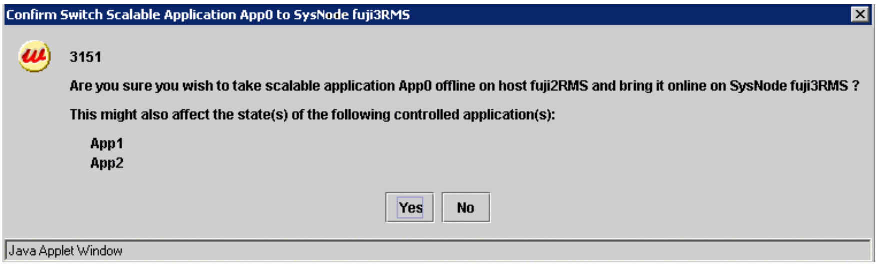
When you select an item in an object's pop-up menu that can cause state changes to that object, a confirmation pop-up window appears (Figure 74). To proceed with the action described in the warning message, click *Yes*; to cancel the action, click *No*.


Confirm Online Application app2

3053
Are you sure you wish to bring application app2 online on SysNode fuji2RMS ?

Figure 74: Confirmation pop-up window

For a scalable `userApplication` object, the confirmation pop-up lists the controlled applications and warns that their states can also change with the specified action (Figure 75).

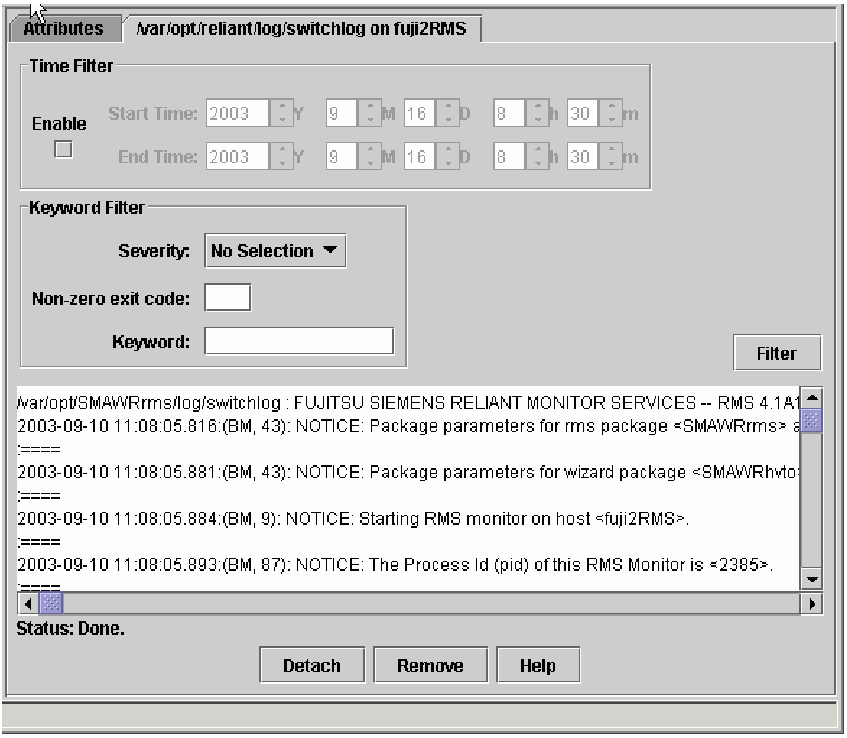

Confirm Switch Scalable Application App0 to SysNode fuji3RMS

3151
Are you sure you wish to take scalable application App0 offline on host fuji2RMS and bring it online on SysNode fuji3RMS ?
This might also affect the state(s) of the following controlled application(s):
App1
App2

Figure 75: Confirmation pop-up window for scalable application

5.2.4.5 Switchlogs and application logs

The switchlog on individual system nodes can be viewed by using the *View Switchlog* option from the system node command pop-up window. The switchlog is displayed in a tab on the right-side panel (Figure 76).



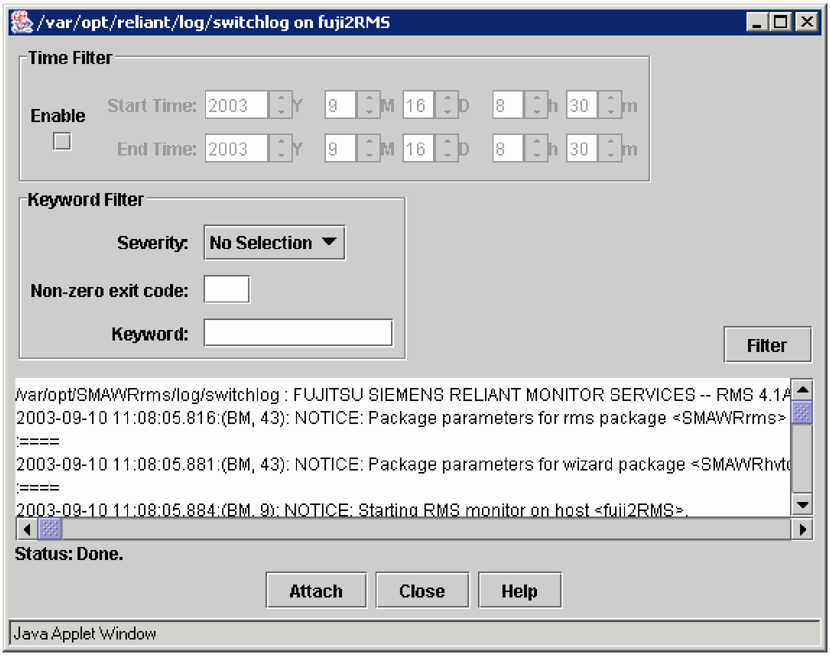
Attributes
/var/opt/reliant/log/switchlog on fuji2RMS
Time Filter
Enable
Start Time: 2003 Y 9 M 16 D 8 h 30 m
End Time: 2003 Y 9 M 16 D 8 h 30 m
Keyword Filter
Severity: No Selection
Non-zero exit code:
Keyword:
Filter
Detach
Remove
Help

```
/var/opt/SMAWRrms/log/switchlog : FUJITSU SIEMENS RELIANT MONITOR SERVICES -- RMS 4.1A1
2003-09-10 11:08:05.816:(BM, 43): NOTICE: Package parameters for rms package <SMAWRrms> a
:====
2003-09-10 11:08:05.881:(BM, 43): NOTICE: Package parameters for wizard package <SMAWRhvto
:====
2003-09-10 11:08:05.884:(BM, 9): NOTICE: Starting RMS monitor on host <fuji2RMS>.
:====
2003-09-10 11:08:05.893:(BM, 87): NOTICE: The Process Id (pid) of this RMS Monitor is <2385>.
:====
```

Status: Done.

Figure 76: Viewing the RMS switchlog file

The *Detach* button will separate the switchlog tab so you can view it in its own window (Figure 77). The detached window can be rejoined to the main window with the *Attach* button.

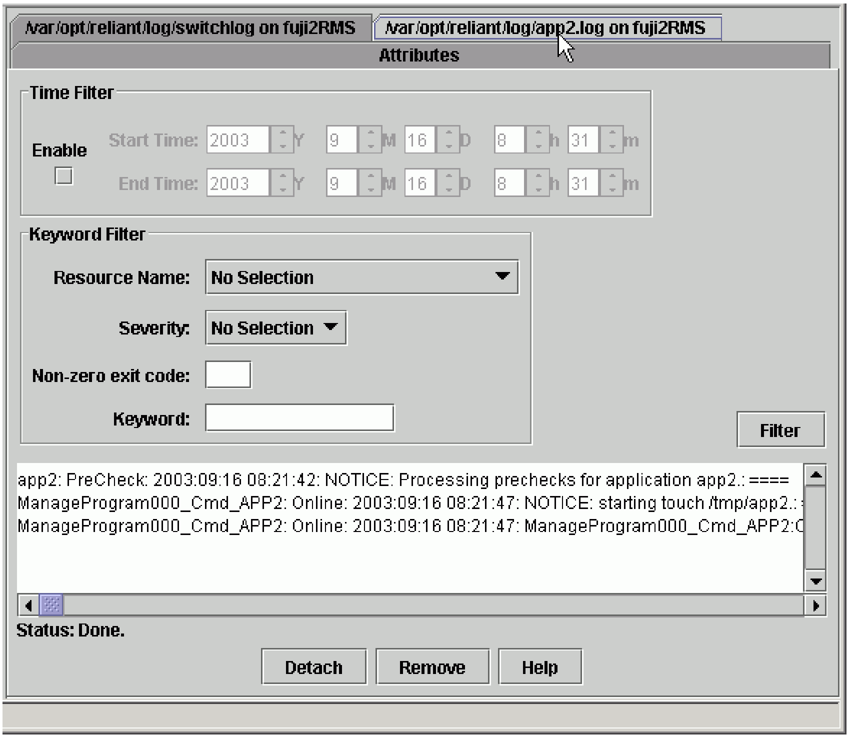


```
/var/opt/SMAWRrms/log/switchlog : FUJITSU SIEMENS RELIANT MONITOR SERVICES -- RMS 4.1A
2003-09-10 11:08:05.816:(BM, 43): NOTICE: Package parameters for rms package <SMAWRrms>
:====
2003-09-10 11:08:05.881:(BM, 43): NOTICE: Package parameters for wizard package <SMAWRhvtc
:====
2003-09-10 11:08:05.884:(BM, 9): NOTICE: Starting RMS monitor on host <fuji2RMS>.
```

Status: Done.

Figure 77: Viewing the RMS switchlog file in a detached window

Display the application log by right-clicking on an online application on the RMS tree and choosing *View Logfile* (Figure 78).


/var/opt/reliant/log/switchlog on fuji2RMS
/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Time Filter
Enable
Start Time: 2003 Y 9 M 16 D 8 h 31 m
End Time: 2003 Y 9 M 16 D 8 h 31 m
Keyword Filter
Resource Name: No Selection
Severity: No Selection
Non-zero exit code:
Keyword:
Filter
Status: Done.
Detach
Remove
Help

```
app2: PreCheck: 2003:09:16 08:21:42: NOTICE: Processing prechecks for application app2.: ====
ManageProgram000_Cmd_APP2: Online: 2003:09:16 08:21:47: NOTICE: starting touch /tmp/app2.:
ManageProgram000_Cmd_APP2: Online: 2003:09:16 08:21:47: ManageProgram000_Cmd_APP2:C
```

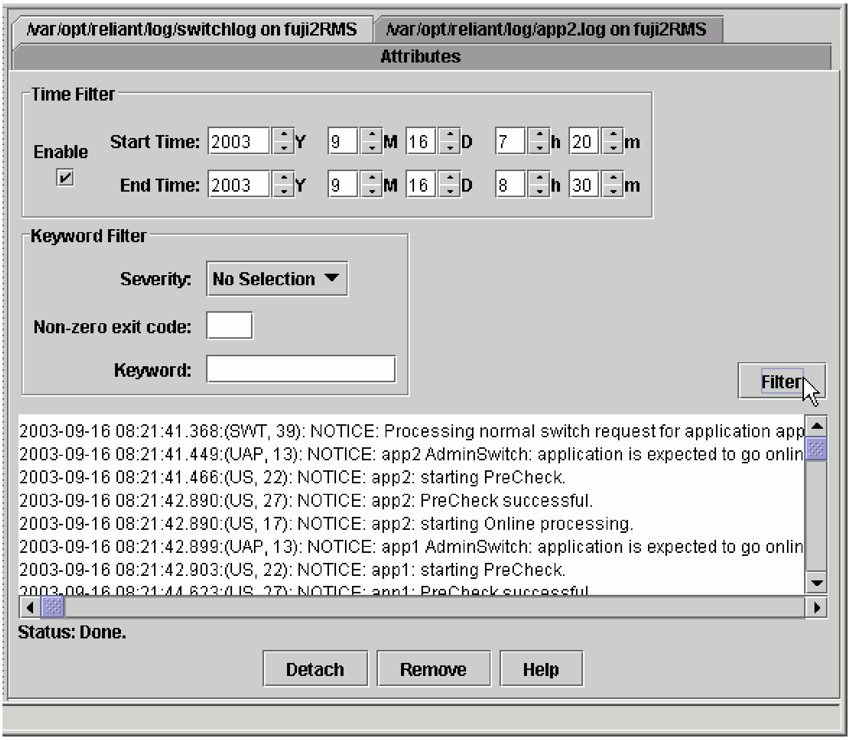
Figure 78: Viewing the application log

By default, the entire log is available in the scrolled area at the bottom of the window. You can restrict the entries displayed with the following filters:

- Timestamp: Click the *Enable* check box and select the period of interest.
- Resource name, 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.

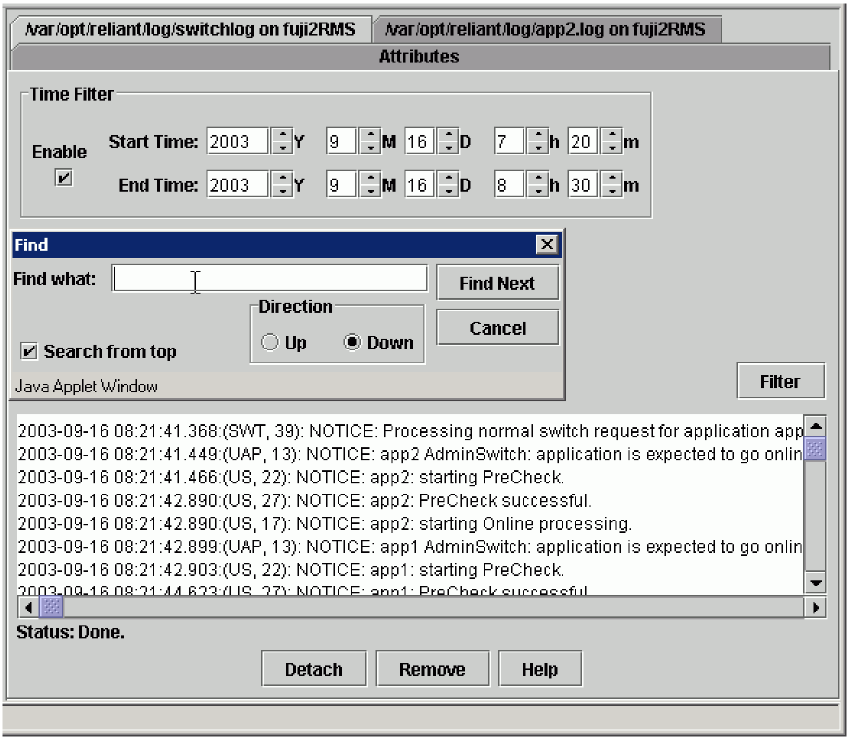
Click the *Filter* button to display the filtered log entries. Figure 79 shows the screen for a search based on the date and time.


/var/opt/reliant/log/switchlog on fuji2RMS
/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Time Filter
Enable
Start Time: 2003 Y 9 M 16 D 7 h 20 m
End Time: 2003 Y 9 M 16 D 8 h 30 m
Keyword Filter
Severity: No Selection
Non-zero exit code:
Keyword:
Filter
Status: Done.
Detach
Remove
Help

```
2003-09-16 08:21:41.368:(SWT, 39): NOTICE: Processing normal switch request for application app
2003-09-16 08:21:41.449:(UAP, 13): NOTICE: app2 AdminSwitch: application is expected to go onlin
2003-09-16 08:21:41.466:(US, 22): NOTICE: app2: starting PreCheck.
2003-09-16 08:21:42.890:(US, 27): NOTICE: app2: PreCheck successful.
2003-09-16 08:21:42.890:(US, 17): NOTICE: app2: starting Online processing.
2003-09-16 08:21:42.899:(UAP, 13): NOTICE: app1 AdminSwitch: application is expected to go onlin
2003-09-16 08:21:42.903:(US, 22): NOTICE: app1: starting PreCheck.
2003-09-16 08:21:44.623:(US, 27): NOTICE: app1: PreCheck successful.
```

Figure 79: Search based on date and time filter

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* option (Figure 80).


/var/opt/reliant/log/switchlog on fuji2RMS
/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Time Filter
Enable ✓
Start Time: 2003 Y 9 M 16 D 7 h 20 m
End Time: 2003 Y 9 M 16 D 8 h 30 m
Find
Find what:
Find Next
Cancel
Direction
Up ● Down
✓ Search from top
Java Applet Window
Filter
Status: Done.
Detach
Remove
Help

```
2003-09-16 08:21:41.368:(SWT, 39): NOTICE: Processing normal switch request for application app
2003-09-16 08:21:41.449:(UAP, 13): NOTICE: app2 AdminSwitch: application is expected to go onlin
2003-09-16 08:21:41.466:(US, 22): NOTICE: app2: starting PreCheck.
2003-09-16 08:21:42.890:(US, 27): NOTICE: app2: PreCheck successful.
2003-09-16 08:21:42.890:(US, 17): NOTICE: app2: starting Online processing.
2003-09-16 08:21:42.899:(UAP, 13): NOTICE: app1 AdminSwitch: application is expected to go onlin
2003-09-16 08:21:42.903:(US, 22): NOTICE: app1: starting PreCheck.
2003-09-16 08:21:44.623:(US, 27): NOTICE: app1: PreCheck successful
```

Figure 80: Using the Find pop-up in log viewer

5.2.5 RMS graphs

Cluster Admin contains the following RMS graphs, which are useful for graphically viewing the details of the RMS configuration file:

- 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.

You can use any graph for access to the following features:

- Configuration information from a graph
- Command pop-ups
- RMS graph customization
- Node status after RMS is shut down

These graphs and their features are explained in more detail in the sections that follow.

5.2.5.1 RMS full graph

The RMS full graph displays the complete configuration of the cluster (Figure 81). The graph represents the following items in the RMS configuration:

- Relationships between objects
- Dependencies of objects
- Object types
- Current node state

You can see the RMS full graph by right-clicking on a system node. The RMS graph is drawn from the perspective of a particular system node; that is, the state information of all the nodes is displayed as viewed from a particular system node. You can view an RMS graph from the perspective of any of the system nodes. The node name in the title bar of the graph identifies the node that is supplying the state information.

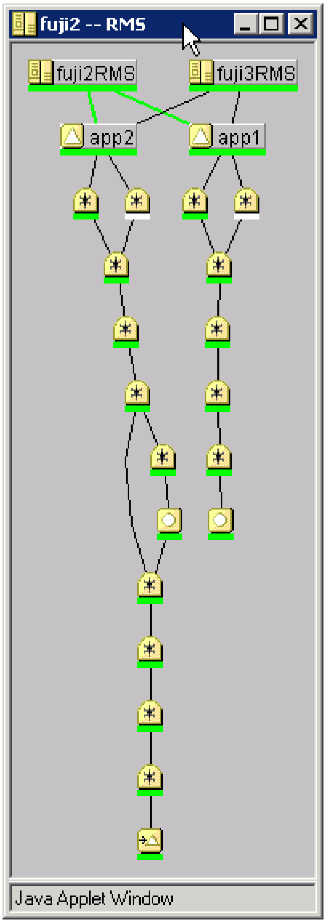


Figure 81: RMS full graph

5.2.5.2 Application graph

You can see a graph for a single application by right-clicking on an application. The application graph shows all the resources used by that specific application. You can also look at specific resource properties. The application graph is similar to the full graph, except that it shows just a single application and its resources. The graph is shown from the perspective of the selected node (Figure 82).

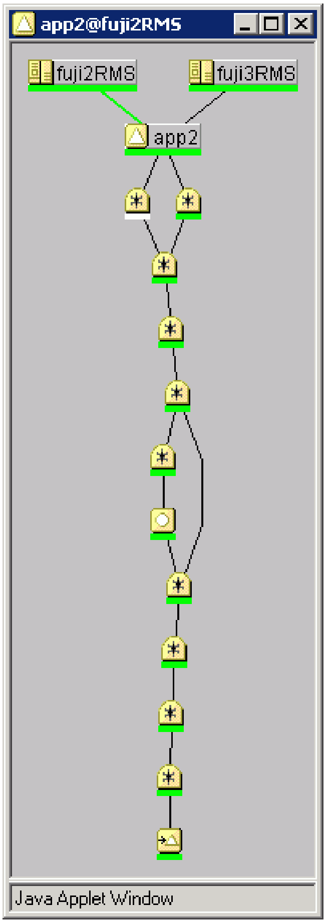


Figure 82: RMS application graph

5.2.5.3 Subapplication graph

You can see a graph for a subapplication by right-clicking on a subapplication.The subapplication graph lists all the subapplications used by a given application, and it shows the connections between the subapplications (Figure 83).

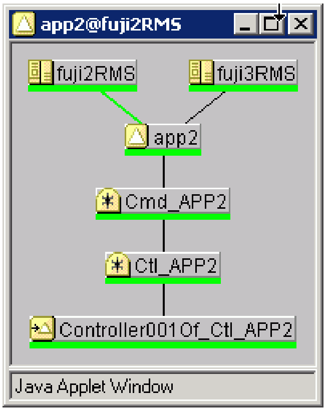


Figure 83: RMS subapplication graph

5.2.5.4 Composite subapplication graph

The composite subapplication graph is a variation of the subapplication graph. If an application has a dependency on another application by means of a controller object, then the composite subapplication graph can be used to show all the subapplications that the application depends on directly or indirectly. For example, a composite graph may depict a Web Server application that depends on an Oracle Database Server application.

The composite subapplication graph takes the controller object in a subapplication graph and appends the subapplication graph of the controlled application below it. This gives a composite view of all the subapplications that the first application depended on directly or indirectly (Figure 84). If the controlled application has further controller objects, then the process is recursively repeated.

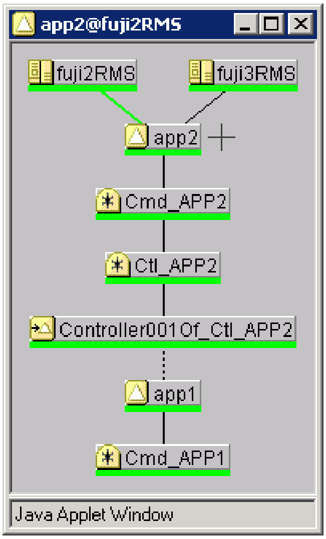
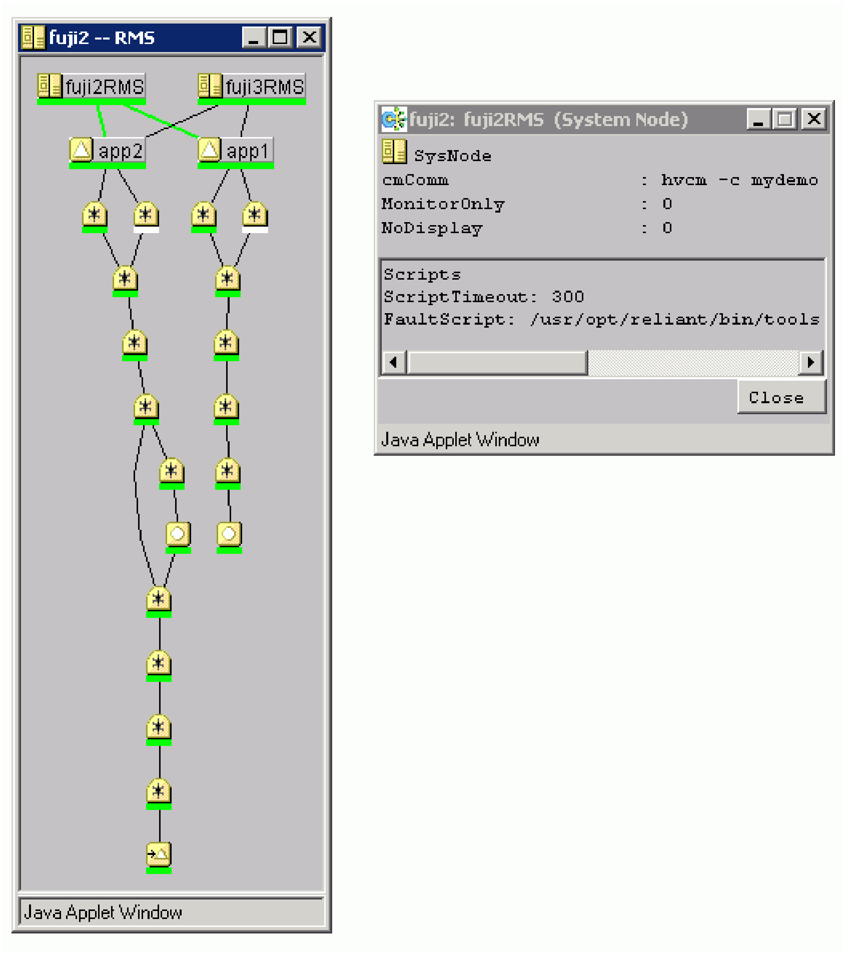


Figure 84: Composite subapplication graph

5.2.5.5 Configuration information from a graph

Click the left mouse button on the object of interest to see the configuration information of the object in a graph form. A pop-up screen displays the attributes (Figure 85).



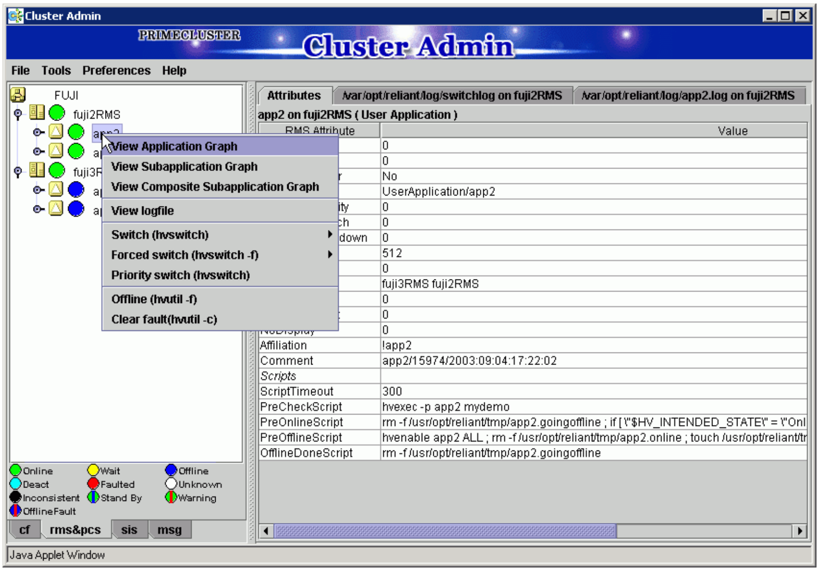
```
SysNode
cmComm                  : hvcm -c mydemo
MonitorOnly             : 0
NoDisplay               : 0
```

```
Scripts
ScriptTimeout: 300
FaultScript: /usr/opt/reliant/bin/tools
```

Figure 85: Configuration information pop-up

5.2.5.6 Command pop-ups

You can use the context sensitive command pop-up menus on the RMS graph nodes to perform many operations. 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 86).



app2 on fuji2RMS (User Application)

RMS Attribute	Value
	0
	0
r	No
	UserApplication/app2
ity	0
ch	0
down	0
	512
	0
	fuji3RMS fuji2RMS
	0
	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onl
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tr
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 86: Command pop-up

5.2.5.7 RMS graph customization

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 87 shows a graph that displays affiliation names.

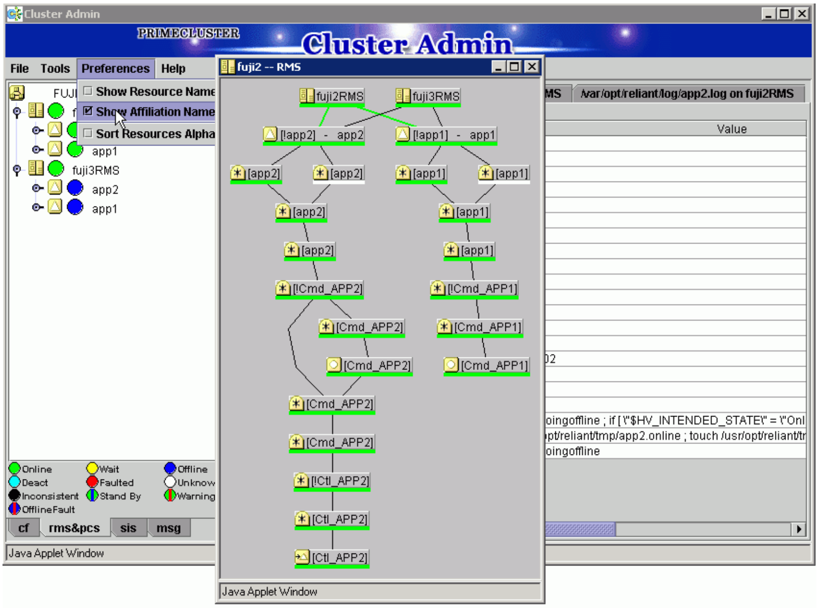


Figure 87: RMS graph with affiliation names

Figure 88 shows a graph that displays resource names.

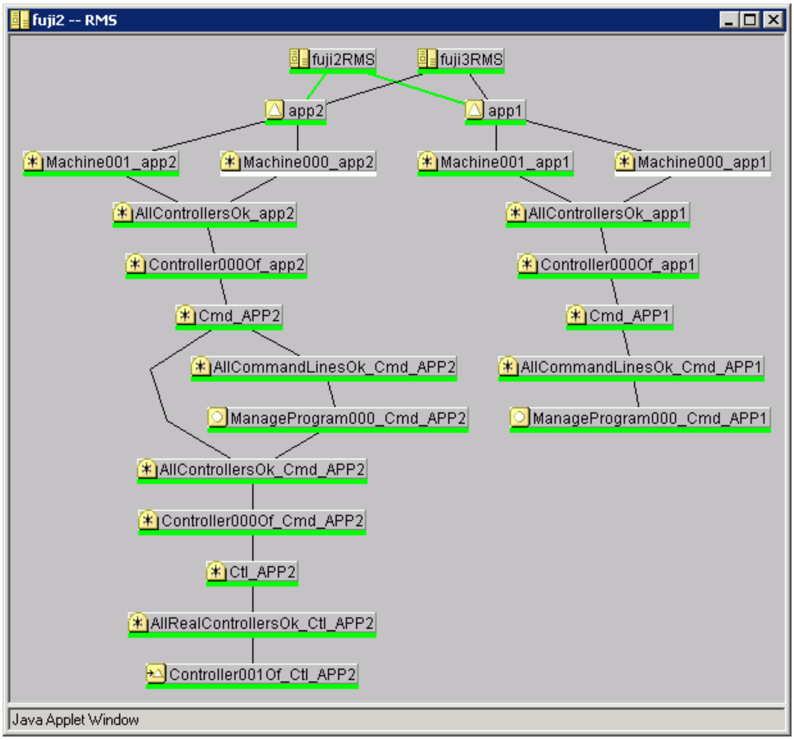
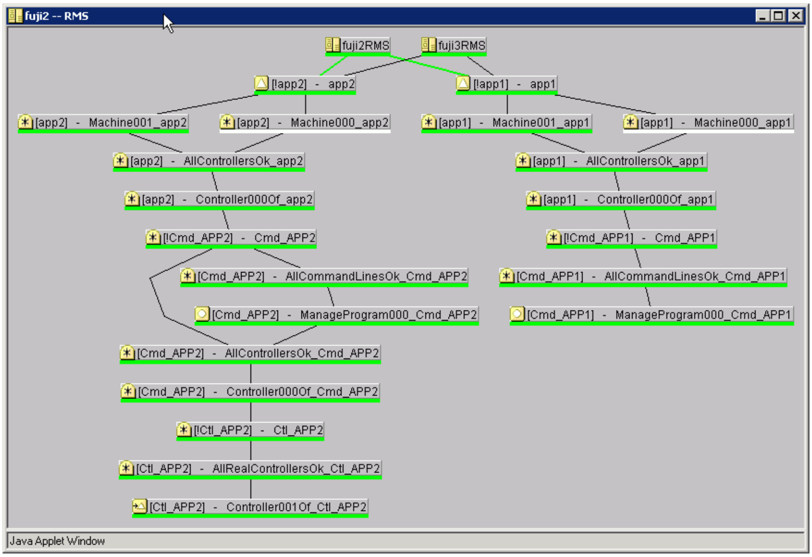


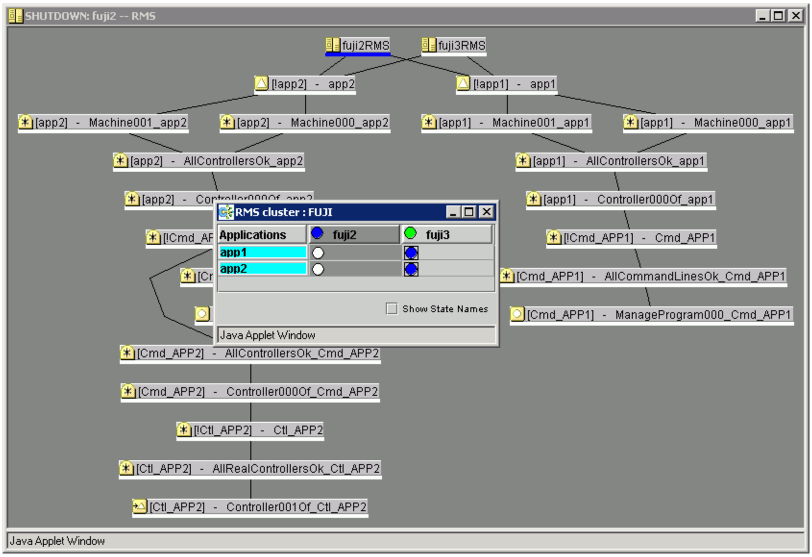
Figure 88: RMS graph with resource names

If both options are selected, graphs will display both the affiliation names and resource names. This combination stretches the graph horizontally and can make it difficult to read (Figure 89).


fuji2 -- RMS
fuji2RMS
fuji3RMS
[!app2] - app2
[!app1] - app1
[app2] - Machine001_app2
[app2] - Machine000_app2
[app1] - Machine001_app1
[app1] - Machine000_app1
[app2] - AllControllersOk_app2
[app1] - AllControllersOk_app1
[app2] - Controller000Of_app2
[app1] - Controller000Of_app1
[!Cmd_APP2] - Cmd_APP2
[!Cmd_APP1] - Cmd_APP1
[Cmd_APP2] - AllCommandLinesOk_Cmd_APP2
[Cmd_APP1] - AllCommandLinesOk_Cmd_APP1
[Cmd_APP2] - ManageProgram000_Cmd_APP2
[Cmd_APP1] - ManageProgram000_Cmd_APP1
[Cmd_APP2] - AllControllersOk_Cmd_APP2
[Cmd_APP2] - Controller000Of_Cmd_APP2
[!Ctl_APP2] - Ctl_APP2
[Ctl_APP2] - AllRealControllersOk_Ctl_APP2
[Ctl_APP2] - Controller001Of_Ctl_APP2
Java Applet Window
Figure 89: RMS graph with affiliation names and resource names

5.2.5.8 Node status after RMS is shut down

After RMS is shut down, the RMS GUI windows become dark gray on the node from which they are getting their information (Figure 90). In this condition, all the states are white, indicating that the states are unknown. The main window and the clusterwide table continue to show the application states until RMS is shut down on all nodes.


SHUTDOWN: fuji2 -- RMS
fuji2RMS
fuji3RMS
[!app2] - app2
[!app1] - app1
[app2] - Machine001_app2
[app2] - Machine000_app2
[app1] - Machine001_app1
[app1] - Machine000_app1
[app2] - AllControllersOk_app2
[app1] - AllControllersOk_app1
[app2] - Controller000Of_app2
[app1] - Controller000Of_app1
[!Cmd_AP
[!Cmd_APP1] - Cmd_APP1
[Cr
[Cmd_APP1] - AllCommandLinesOk_Cmd_APP1
[Cmd_APP1] - ManageProgram000_Cmd_APP1
[Cmd_APP2] - AllControllersOk_Cmd_APP2
[Cmd_APP2] - Controller000Of_Cmd_APP2
[!Ctl_APP2] - Ctl_APP2
[Ctl_APP2] - AllRealControllersOk_Ctl_APP2
[Ctl_APP2] - Controller001Of_Ctl_APP2
RMS cluster : FUJI
Show State Names
Java Applet Window
Java Applet Window

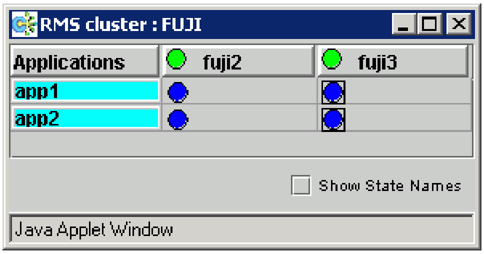
Applications	fuji2	fuji3
app1	○	●
app2	○	●

Figure 90: RMS graph after RMS is shut down

5.2.6 RMS clusterwide table

The RMS clusterwide table displays the state information about `userApplication` objects as a summary table. The user can see the state of each of the `userApplication` objects on each of the system nodes. It presents the information in a concise manner.

Open the clusterwide table through a pop-up menu option for the cluster node (root node) in the RMS tree. The clusterwide table comes up in a separate window (Figure 91).


RMS cluster : FUJI

Applications	● fuji2	● fuji3
app1	●	●
app2	●	●

☐ Show State Names

Java Applet Window

Figure 91: Clusterwide table

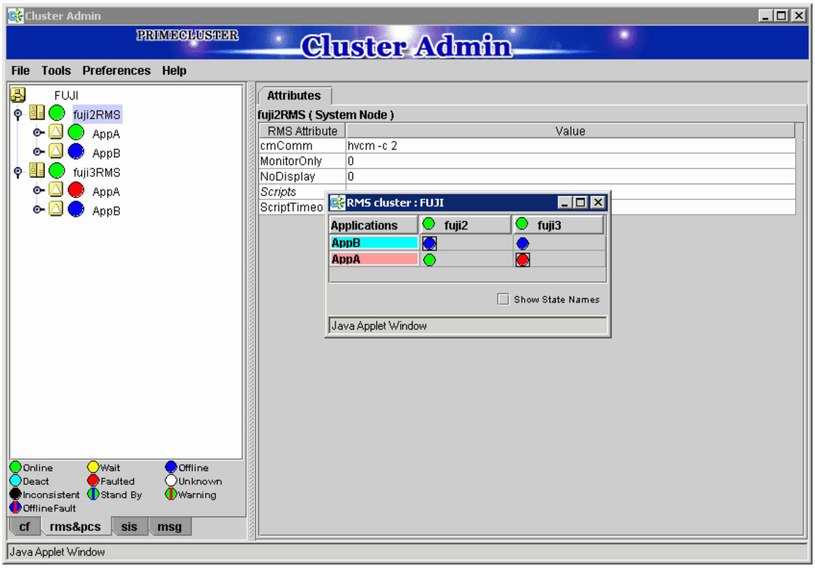
You can increase or decrease the size of the clusterwide table window and the size of the columns by using the mouse. If the window is already large enough to fully display all of the table elements, then you will not be allowed to further increase its size.

A square surrounding the colored state circle indicates the primary node for the application. Figure 91 shows that `fuji2` 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. This allows the System Administrator to easily spot any `Faulted` applications.

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, and the application's name is highlighted in a light blue. Thus, the System Administrator can see what applications are not running anywhere and should probably be brought online on some node.

If there are both `Faulted` applications and applications that are not online anywhere, then the `Faulted` applications are shown above the ones that are not online anywhere.

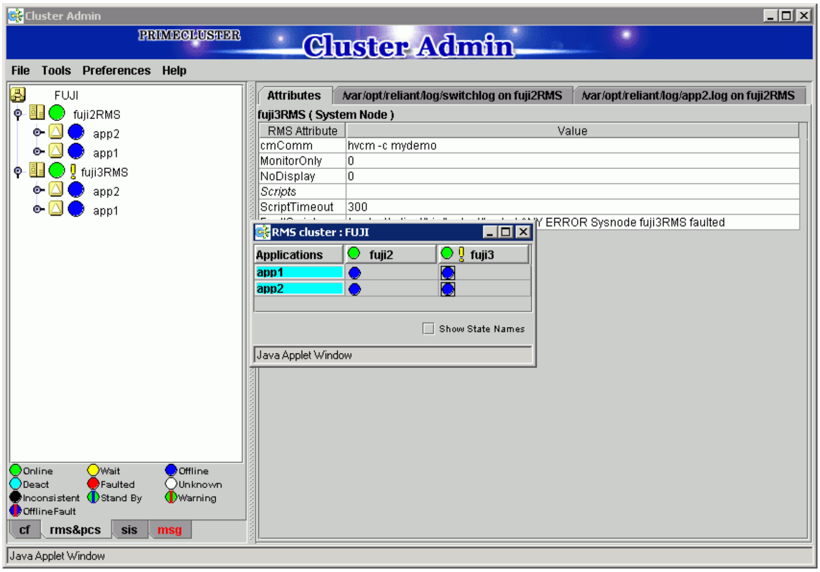


RMS Attribute	Value
cmComm	hvcm -c 2
MonitorOnly	0
NoDisplay	0
Scripts	
ScriptTimeo	

Applications	fuji2	fuji3
AppB	●	●
AppA	●	●

Figure 92: 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 colored circles for `SysNode`s. 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 93 shows a clusterwide table with an application of a split-brain condition.



fuji3RMS (System Node)

RMS Attribute	Value
cmComm	hvcm -c mydemo
MonitorOnly	0
NoDisplay	0
Scripts	
ScriptTimeout	300
	Y ERROR Sysnode fuji3RMS faulted

Applications	● fuji2	● ! fuji3
app1	●	●
app2	●	●

Figure 93: Exclamation marks in clusterwide table and the RMS tree

5.2.6.1 Command pop-ups

Use the context-sensitive command pop-up menus to perform some of the operations on the clusterwide table nodes. 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 node (Figure 94).

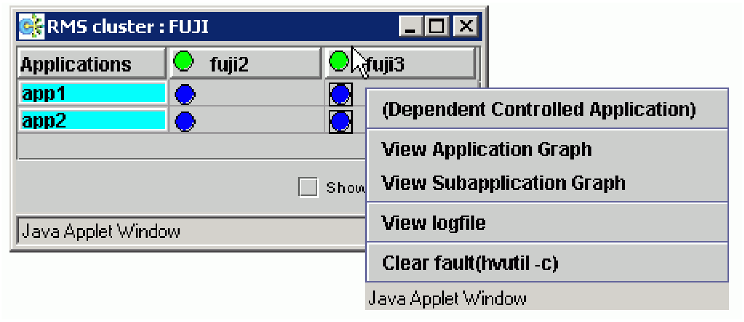
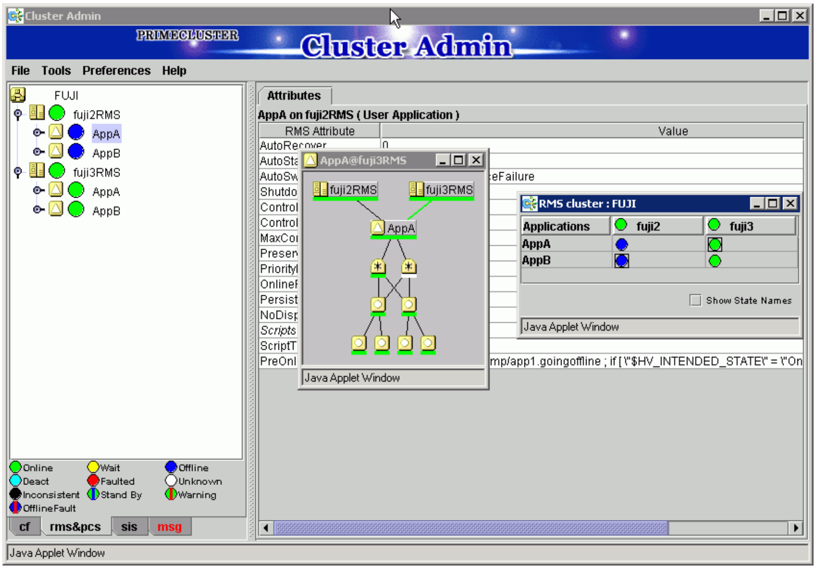


Figure 94: Command pop-ups in clusterwide table

5.2.7 Changing the RMS configuration

When you stop and restart RMS with a different configuration, the graphs, 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 95 illustrates the display containing `AppA` and `AppB` before RMS is shutdown, and Figure 96 shows the RMS GUI after RMS has been restarted with a different configuration that uses `app1` and `app2`.

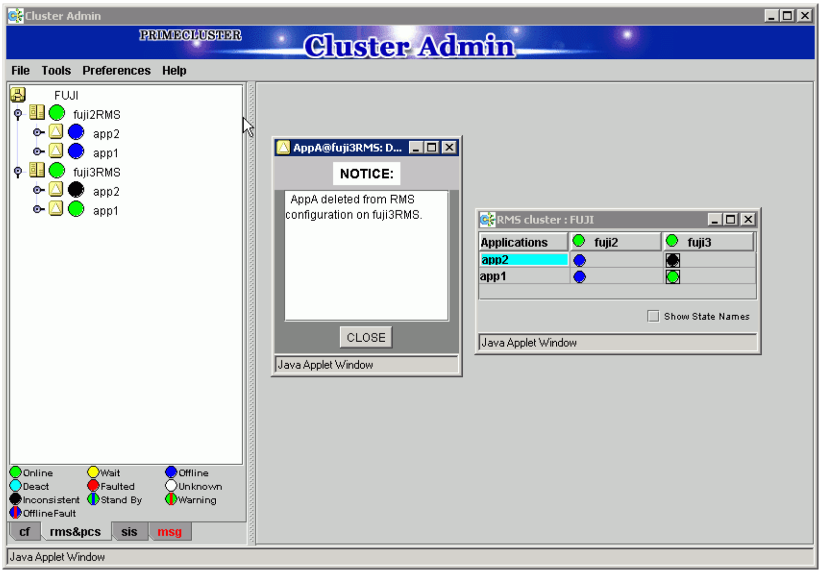


AppA on fuji2RMS (User Application)

RMS Attribute	Value
AutoRecover	0
AutoSta	
AutoSw	eFailure
Shutdo	
Control	
Control	
MaxCo	
Presen	
Priority	
OnlineF	
Persist	
NoDisp	
Scripts	
ScriptT	
PreOnl	mp/app1.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"On

Applications	fuji2	fuji3
AppA	●	●
AppB	●	●

Figure 95: Before RMS is shut down



NOTICE:

AppA deleted from RMS configuration on fuji3RMS.

Applications	fuji2	fuji3
app2	●	●
app1	●	●

Figure 96: After RMS is restarted with a different configuration

5.3 RMS procedures

Each of the following sections presents two alternative methods:

- GUI—The Cluster Admin interface is the preferred method of operation.
- CLI—The commands here employ the most commonly used options. For more details about any command, see the online manual pages, which are listed in the chapter “Appendix—List of manual pages” on page 349. The commands are located in the *RELIANT_PATH*`/bin` directory.

  i All the RMS CLI commands accept both CF node names and RMS node names for `SysNode` objects when the RMS naming convention is followed (that is, the names are of the form *nodename*RMS).

5.3.1 Starting RMS

1. From the Cluster Admin *rms* tab, select *Tools > Start RMS* (Figure 97).

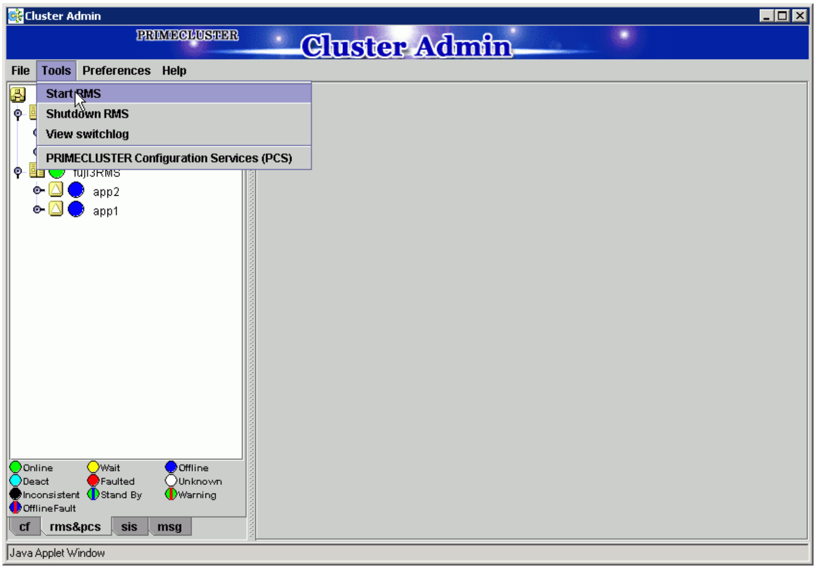
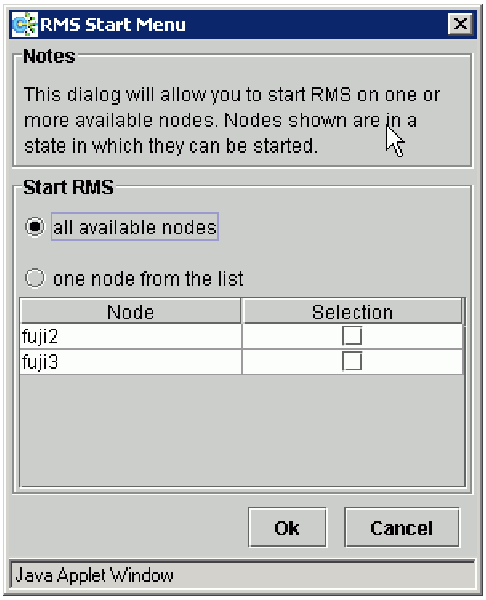


Figure 97: Starting RMS from the main menu

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 98).


RMS Start Menu

Notes

This dialog will allow you to start RMS on one or more available nodes. Nodes shown are in a state in which they can be started.

Start RMS

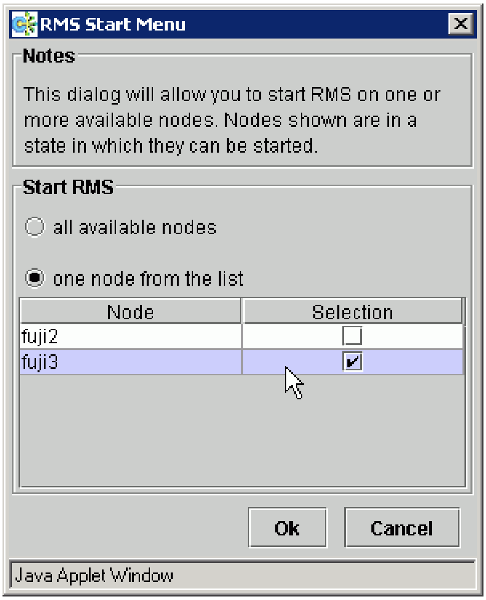
[✓ all available nodes]

one node from the list

Node	Selection
fuji2	☐
fuji3	☐

Figure 98: RMS Start Menu for all nodes

3. To start RMS only on selected nodes, click the *one node from the list* radio button; select the desired node or nodes using the checkboxes in the *Selection* column; and then click *OK* (Figure 99).



RMS Start Menu

Notes

This dialog will allow you to start RMS on one or more available nodes. Nodes shown are in a state in which they can be started.

Start RMS

( ) all available nodes

(●) one node from the list

Node	Selection
fuji2	☐
fuji3	✓

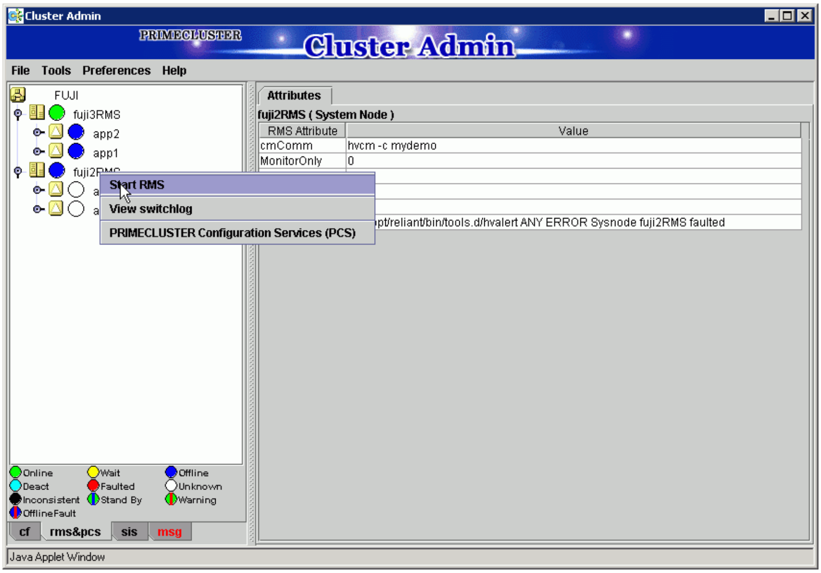
Ok Cancel

Java Applet Window

Figure 99: RMS Start Menu for individual nodes

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

1. In the left pane, click the *rms* tab to view the cluster tree.
2. Right-click on the node and select *StartRMS* from the pop-up menu (Figure 100).



RMS Attribute	Value
cmComm	hvcm -c mydemo
MonitorOnly	0
	pt/reliant/bin/tools.d/hvalert ANY ERROR Sysnode fuji2RMS faulted

Figure 100: Starting RMS on individual nodes

CLI

The syntax for the CLI is as follows:

`hvcm [-c` *config_file*`] {-a | -s` *SysNode*`}`

The `hvcm` command starts RMS with the configuration file specified by the `-c` option. If no `-c` is present, RMS uses the default startup file `CONFIG.rms`.

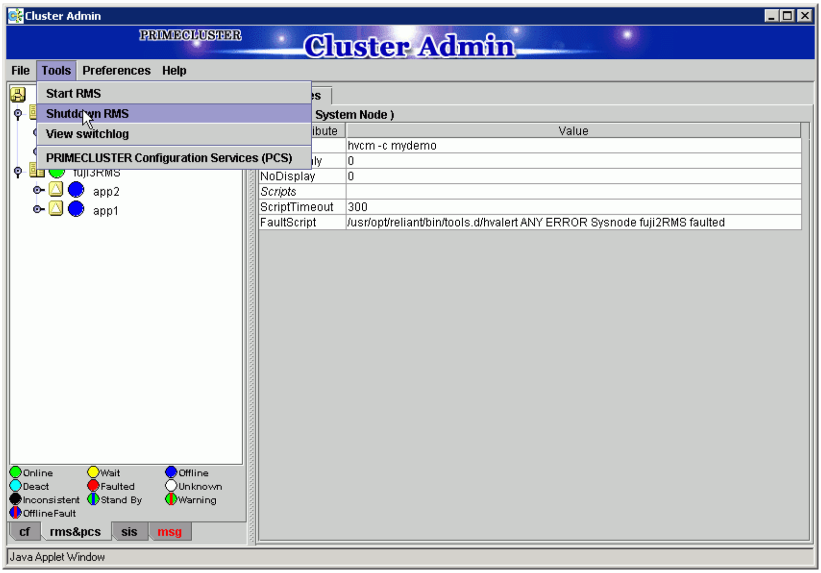
The `hvcm` command starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the `hvcm` command; the default values are sufficient for most configurations.

The default startup file, `CONFIG.rms`, 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 the default root directory `/opt/SMAW/SMAWRrms/etc`.

i The system default run level in `/etc/inittab` must match the chosen RMS start run level; otherwise, the start sequence may be out of order. To verify or to change the RMS run level use the `hvrclev` command. Refer to the chapter "Appendix—List of manual pages" on page 349 for more information.

5.3.2 Stopping RMS

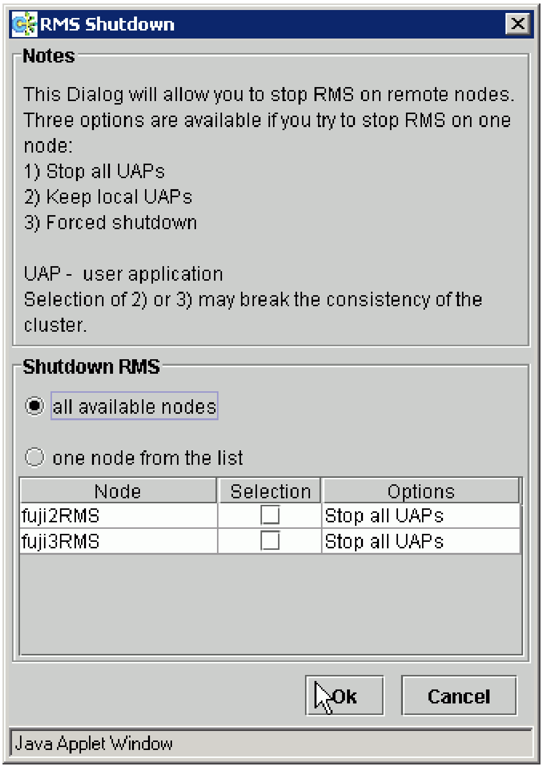
1. Use the *Tools* pull-down menu, or right-click on a system node, and select the mode of shutdown in the subsequent option screen (Figure 101).



ibute	Value
	hvcm -c mydemo
ly	0
NoDisplay	0
Scripts	
ScriptTimeout	300
FaultScript	/usr/opt/reliant/bin/tools.d/hvalert ANY ERROR Sysnode fuji2RMS faulted

Figure 101: Stopping RMS

2. Select the radio button for *all available nodes* and click *Ok* to shutdown RMS on all nodes (Figure 102).


RMS Shutdown

Notes

This Dialog will allow you to stop RMS on remote nodes. Three options are available if you try to stop RMS on one node:
1) Stop all UAPs
2) Keep local UAPs
3) Forced shutdown

UAP - user application
Selection of 2) or 3) may break the consistency of the cluster.

Shutdown RMS

● all available nodes

○ one node from the list

Node	Selection	Options
fuji2RMS	☐	Stop all UAPs
fuji3RMS	☐	Stop all UAPs

Figure 102: Stopping RMS on all available nodes

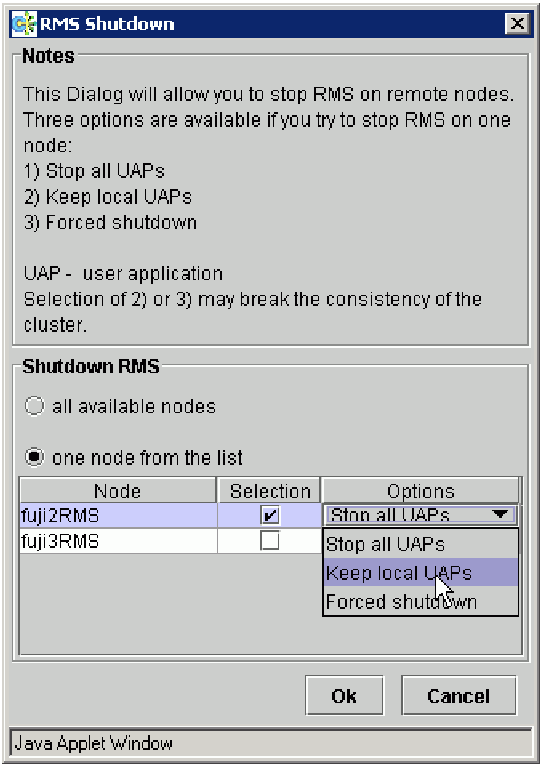
3. To shut down RMS on specific nodes, select the radio button for *one node from the list*, and then click the checkboxes of the nodes you want to shut down (Figure 103). Each node has a dropdown list in the *Options* column to provide additional control:

   - *Stop all UAPs*—Stops all user applications for the selected node
   - *Keep local UAPs*—Leaves the applications running on the selected node
   - *Forced shutdown*—Performs a forced shutdown of RMS

   Caution

   Using a forced shutdown or leaving the applications running and stopping RMS can lead to data inconsistencies or corruption.

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


RMS Shutdown

Notes

This Dialog will allow you to stop RMS on remote nodes. Three options are available if you try to stop RMS on one node:
1) Stop all UAPs
2) Keep local UAPs
3) Forced shutdown

UAP - user application
Selection of 2) or 3) may break the consistency of the cluster.

Shutdown RMS

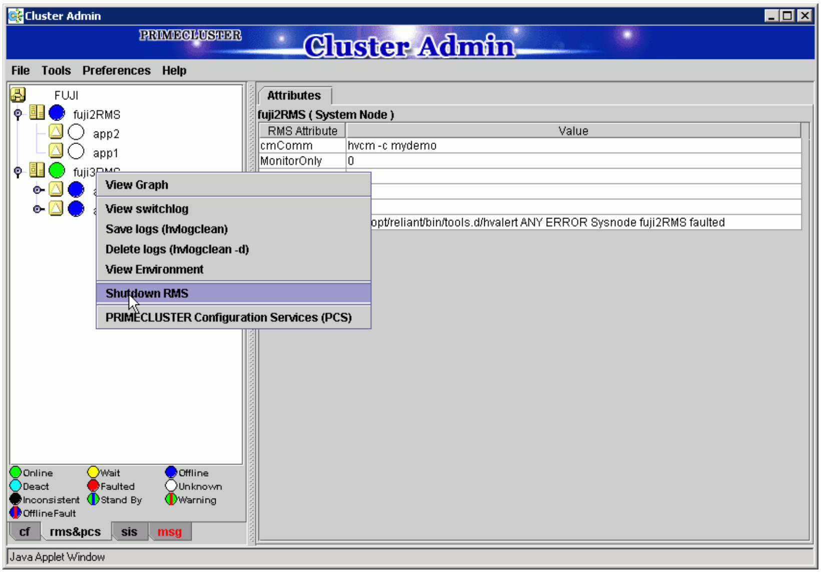
( ) all available nodes

(●) one node from the list

Node	Selection	Options
fuji2RMS	✓	Stop all UAPs
fuji3RMS	☐	

Figure 103: Stopping RMS on one node from the list

Figure 104 shows the command pop-up option to stop RMS on an individual node when you right-click on a system node and select *Shutdown RMS*.



RMS Attribute	Value
cmComm	hvcm -c mydemo
MonitorOnly	0
	opt/reliant/bin/tools.d/hvalert ANY ERROR Sysnode fuji2RMS faulted

Figure 104: Using command pop-up to stop RMS

CLI

The syntax for the CLI is as follows:

`hvshut {-f | -L | -a | -l | -s` *nodename*`}`

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 `-f` or `-L` options are used, the applications remain running but are no longer monitored by RMS.

The `-L` option does a forced shutdown of RMS without shutting down the application. The `-f` option does an emergency shutdown of RMS. Both options only affect the local node, but the `-f` option is for emergencies (when other `hvshut` options do not work).

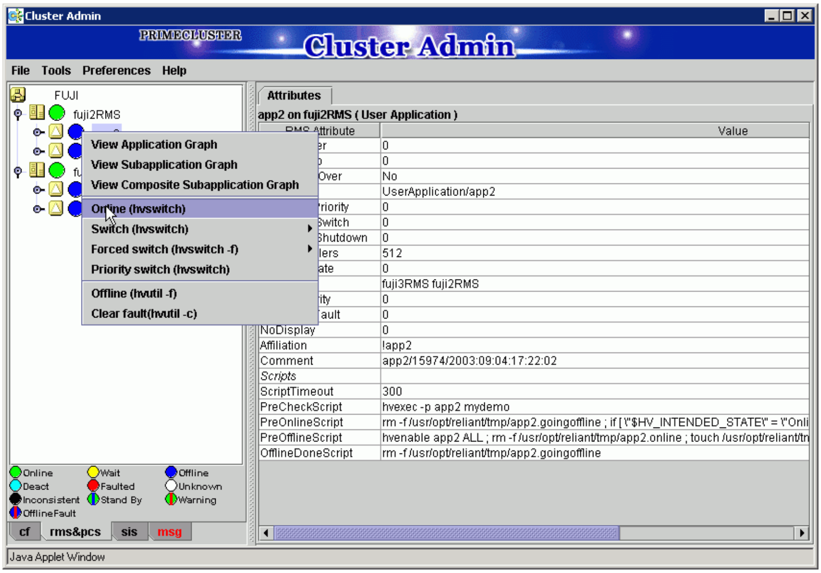
Caution

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

5.3.3 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 105).



app2 on fuji2RMS (User Application)

RMS Attribute	Value
er	0
o	0
Over	No
	UserApplication/app2
riority	0
Switch	0
Shutdown	0
lers	512
ate	0
	fuji3RMS fuji2RMS
rity	0
ault	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 105: Starting an application

CLI

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`. The `-f` option is a forced-switch option.

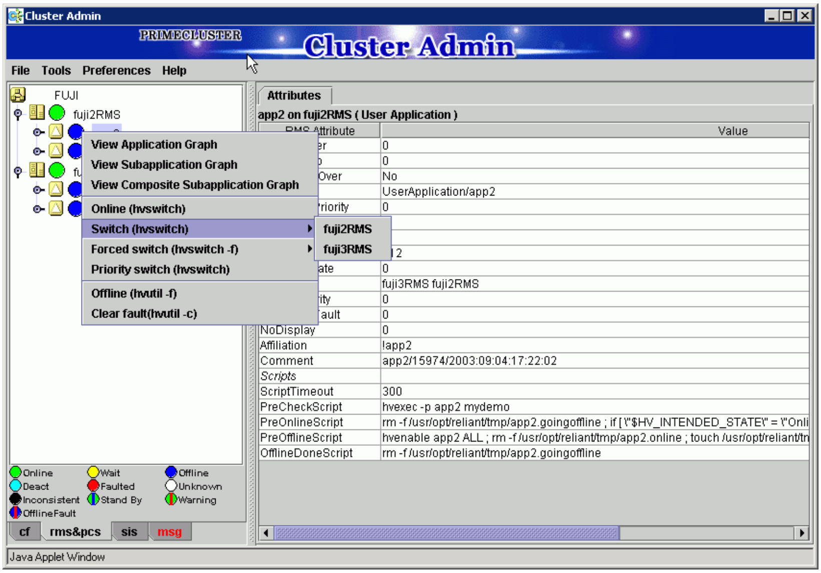
Caution

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

5.3.4 Switching an application

Switch an online application as follows:

1. Right-click on the application object and select the *Switch* menu option. A pull-down menu appears listing the available nodes for switchover.
2. Select the target node from the pull-down menu to switch the application to that node (Figure 106).



app2 on fuji2RMS (User Application)

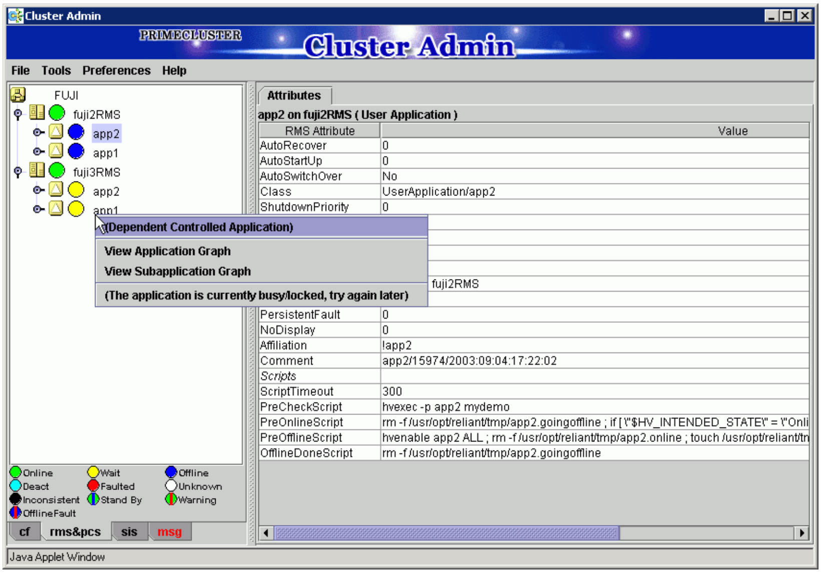
RMS Attribute	Value
er	0
p	0
Over	No
	UserApplication/app2
riority	0
	2
ate	0
	fuji3RMS fuji2RMS
ity	0
ault	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 106: Switching an application

Caution

It is recommended that you use the normal mode of switching applications to ensure that application and data consistencies and integrity are maintained. If an application cannot be switched normally, you may use the forced switch mode; however, a forced 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 107).



(Dependent Controlled Application)
View Application Graph
View Subapplication Graph
(The application is currently busy/locked, try again later)

app2 on fuji2RMS (User Application)

RMS Attribute	Value
AutoRecover	0
AutoStartUp	0
AutoSwitchOver	No
Class	UserApplication/app2
ShutdownPriority	0
	fuji2RMS
PersistentFault	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 107: Switching a busy application

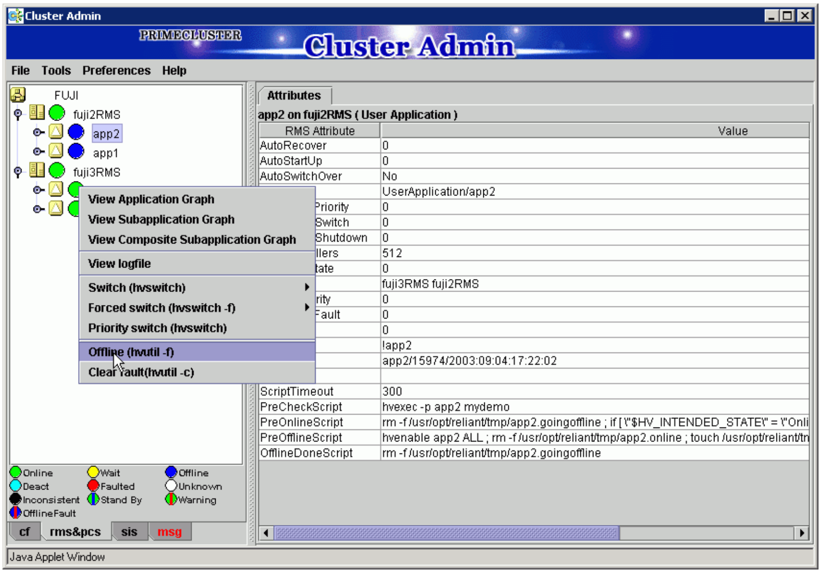
CLI

Refer to the section “Starting an application” on page 134 for information on this command.

5.3.5 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 108).



app2 on fuji2RMS (User Application)

RMS Attribute	Value
AutoRecover	0
AutoStartUp	0
AutoSwitchOver	No
	UserApplication/app2
Priority	0
Switch	0
Shutdown	0
llers	512
tate	0
	fuji3RMS fuji2RMS
rity	0
Fault	0
	0
	!app2
	app2/15974/2003:09:04:17:22:02
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 108: Shutting down an application

CLI

The syntax for the CLI is as follows:

`hvutil -f` *userApplication*

Use the command `hvutil -s` *userApplication* to bring an offline `userApplication` to a `Standby` state.

5.3.6 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

The syntax for the CLI is as follows:

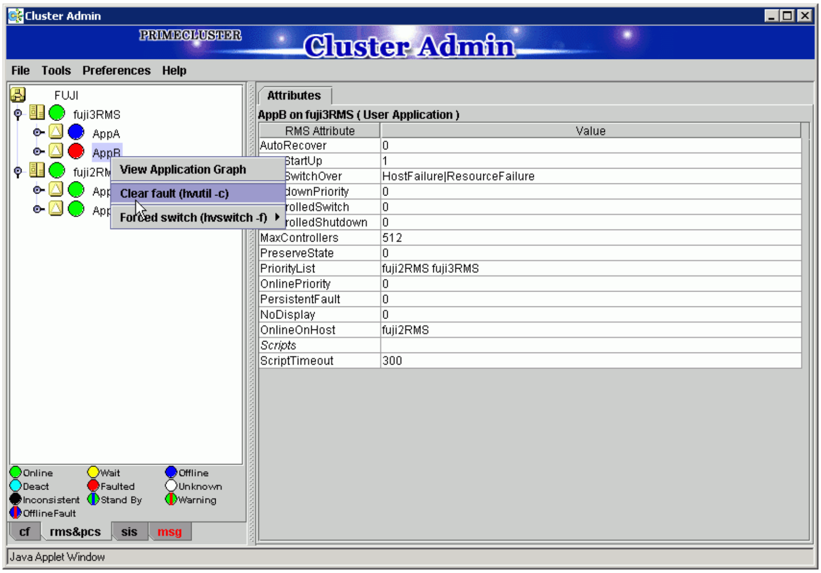
► `hvutil -a userApplication`

i You will not need to activate an application unless someone explicitly deactivated it with the following command: `hvutil -d` *userApplication*.

5.3.7 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 109).



AppB on fuji3RMS (User Application)

RMS Attribute	Value
AutoRecover	0
StartUp	1
SwitchOver	HostFailure|ResourceFailure
downPriority	0
rolledSwitch	0
rolledShutdown	0
MaxControllers	512
PreserveState	0
PriorityList	fuji2RMS fuji3RMS
OnlinePriority	0
PersistentFault	0
NoDisplay	0
OnlineOnHost	fuji2RMS
Scripts	
ScriptTimeout	300

Figure 109: Clearing an application fault

CLI

The syntax for the CLI is as follows:

`hvutil -c` *userApplication*

i If the `userApplication` is in the online state, then clearing the fault will cause RMS to attempt to bring the faulted resource to the online state. If the `userApplication` is in the offline state, then clearing the fault will bring the resource to the offline state.

5.3.8 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

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.

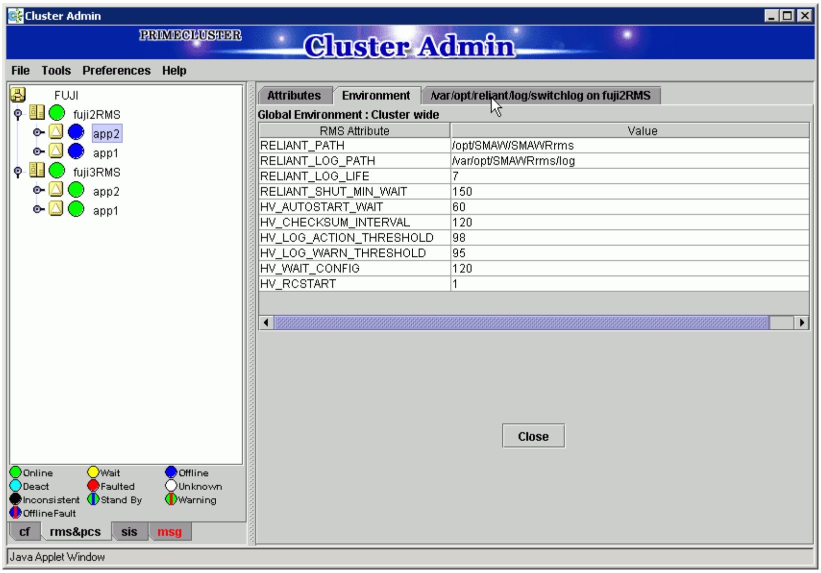
Caution

Manually clearing the `SysNode Wait` state by using either `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.

5.3.9 Displaying environment variables

Display the global or clusterwide environment variables as follows:

► Right-click on a cluster in the RMS tree window and select *View Environment* (Figure 110).



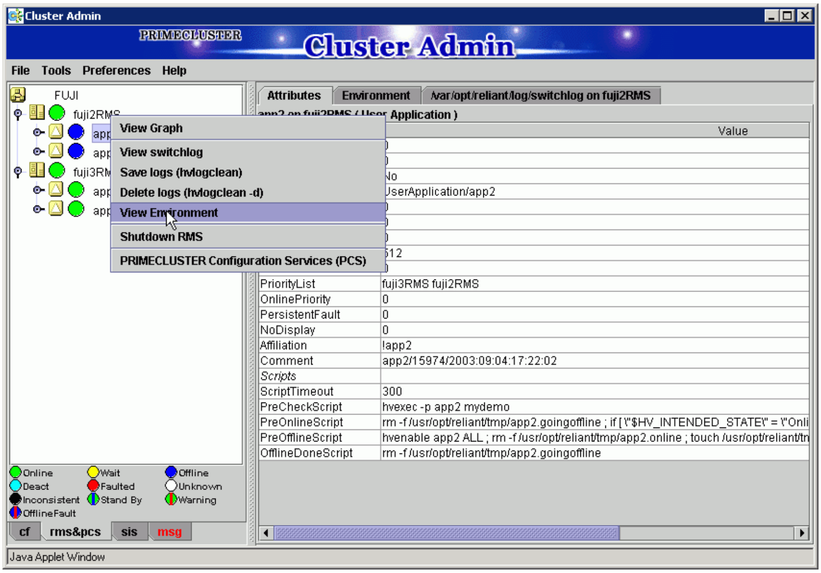
Global Environment : Cluster wide

RMS Attribute	Value
RELIANT_PATH	/opt/SMAW/SMAWRrms
RELIANT_LOG_PATH	/var/opt/SMAWRrms/log
RELIANT_LOG_LIFE	7
RELIANT_SHUT_MIN_WAIT	150
HV_AUTOSTART_WAIT	60
HV_CHECKSUM_INTERVAL	120
HV_LOG_ACTION_THRESHOLD	98
HV_LOG_WARN_THRESHOLD	95
HV_WAIT_CONFIG	120
HV_RCSTART	1

Figure 110: Clusterwide environment variables

Display local environment variables as follows:

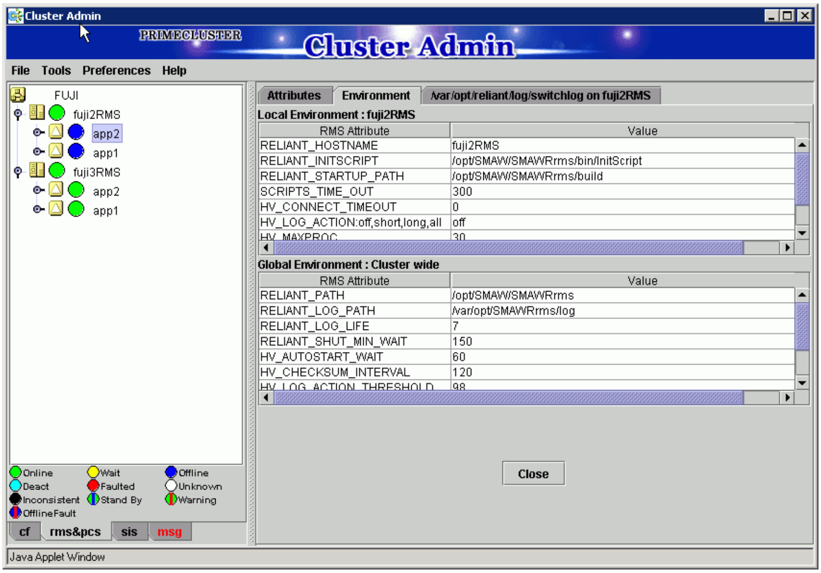
► Right-click on a node in the RMS tree window and select *View Environment* in the command pop-up (Figure 111).



	Value
PriorityList	fuji3RMS fuji2RMS
OnlinePriority	0
PersistentFault	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 111: Local environmental variables pop-up

i Displaying the local environment variables displays the clusterwide environment variables as well (Figure 112).


Cluster Admin

PRIMECLUSTER Cluster Admin

File Tools Preferences Help

- FUJI
  - fuji2RMS
    - app2
    - app1
  - fuji3RMS
    - app2
    - app1

Attributes | Environment | /var/opt/reliant/log/switchlog on fuji2RMS

Local Environment : fuji2RMS

RMS Attribute	Value
RELIANT_HOSTNAME	fuji2RMS
RELIANT_INITSCRIPT	/opt/SMAW/SMAWRrms/bin/InitScript
RELIANT_STARTUP_PATH	/opt/SMAW/SMAWRrms/build
SCRIPTS_TIME_OUT	300
HV_CONNECT_TIMEOUT	0
HV_LOG_ACTION:off,short,long,all	off
HV_MAXPROC	30

Global Environment : Cluster wide

RMS Attribute	Value
RELIANT_PATH	/opt/SMAW/SMAWRrms
RELIANT_LOG_PATH	/var/opt/SMAWRrms/log
RELIANT_LOG_LIFE	7
RELIANT_SHUT_MIN_WAIT	150
HV_AUTOSTART_WAIT	60
HV_CHECKSUM_INTERVAL	120
HV_LOG_ACTION_THRESHOLD	98

🟢 Online	🟡 Wait	🔵 Offline
Deact	🔴 Faulted	○ Unknown
● Inconsistent	Stand By	Warning
OfflineFault		

cf | rms&pcs | sis | msg

Close

Java Applet Window

Figure 112: Local environmental variables window

CLI

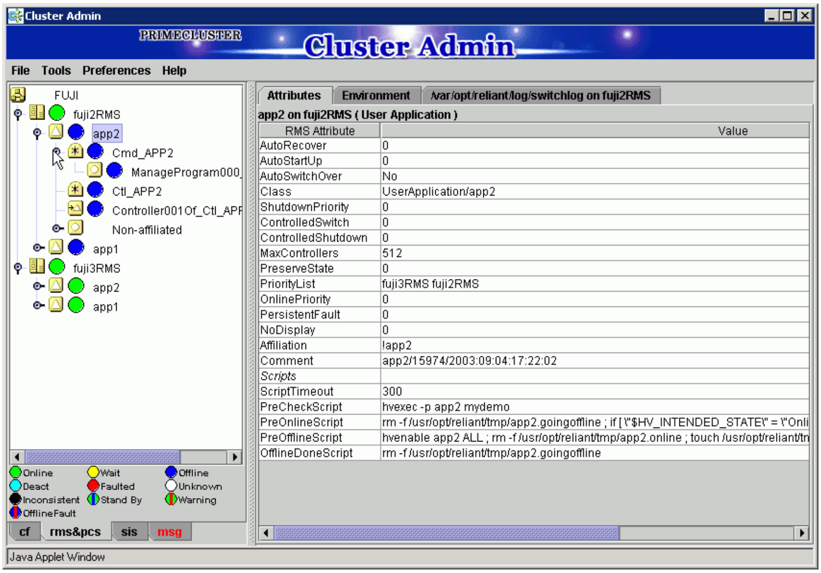
Display the environment variables with the `hvdisp` command, which does not require root privilege:

```
hvdisp ENV
```

```
hvdisp ENVL
```

5.3.10 Displaying application states

The application states of various applications are indicated by different colors. The legend for the application states appears in the RMS main window below the *RMS Tree* panel (Figure 113).



app2 on fuji2RMS (User Application)

RMS Attribute	Value
AutoRecover	0
AutoStartUp	0
AutoSwitchOver	No
Class	UserApplication/app2
ShutdownPriority	0
ControlledSwitch	0
ControlledShutdown	0
MaxControllers	512
PreserveState	0
PriorityList	fuji3RMS fuji2RMS
OnlinePriority	0
PersistentFault	0
NoDisplay	0
Affiliation	!app2
Comment	app2/15974/2003:09:04:17:22:02
Scripts	
ScriptTimeout	300
PreCheckScript	hvexec -p app2 mydemo
PreOnlineScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline ; if [\"$HV_INTENDED_STATE\" = \"Onli
PreOfflineScript	hvenable app2 ALL ; rm -f /usr/opt/reliant/tmp/app2.online ; touch /usr/opt/reliant/tn
OfflineDoneScript	rm -f /usr/opt/reliant/tmp/app2.goingoffline

Figure 113: Displaying application states

CLI

The syntax for the CLI is as follows:

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

The `-a` option displays the `resource_name`, `resource_type`, `HostName` attribute for each resource in the configuration. The `-c` options displays all information in compact format. The `-o` *out_file* option is used to send the output to a file called *out_file*. The `hvdisp` command only works when RMS is running and does not require root privilege.

5.3.11 Viewing the switchlog

View the switchlog for a system node as follows:

- ► Right-click on the system node and select the *View Switchlog* option from the pop-up menu. For more details, refer to the *RMS Troubleshooting Guide*.

You may search the logs based on keywords, date/time ranges, severity levels, or exit codes using the log viewer.

CLI

You can view the switchlog file `/var/opt/SMAWRrms/log/switchlog` using a standard UNIX editor like vi. The *RMS Troubleshooting Guide* describes the RMS log files and their contents.

5.3.12 Viewing application logs

View the application logs as follows:

- ► Right-click on an application on the RMS tree and choose *View logfile* (for more details, refer to the *RMS Troubleshooting Guide*).

5.3.13 Viewing GUI messages

The *Messages* panel displays error and debug messages related to Cluster Admin. View these messages as follows:

- ► Select the *msg* tab on the bottom of the RMS tree panel. This tab turns red if a new message has been added to the text area since it was last viewed.

i Message text area can be cleared or detached from the main panel.

6 Advanced RMS concepts

This chapter deals with ongoing RMS operations, and provides information on RMS runtime behavior, particularly in cases where monitored components fail.

This chapter discusses the following:

- The section “Internal organization” on page 147 briefly describes the object-oriented internal aspects of the RMS base monitor.
- The section “States and scripts” on page 150 lists the RMS scripts.
- The section “Initializing” on page 151 describes the process of transferring the control of nodes to RMS.
- The section “Online processing” on page 152 details the transition of a node to the `Online` state.
- The section “Offline processing” on page 157 details the transition of a node to the `Offline` state.
- The section “Fault processing” on page 159 explains how RMS handles fault situations.
- The section “Switch processing” on page 165 describes how RMS switches applications to other hosts in the cluster.

6.1 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.

6.1.1 Configuration structure

The following rules apply to RMS configurations:

- There must be a `SysNode` object for every node (host) in the cluster.
- A `UserApplication` object is a child of every `SysNode` on which it may run. Therefore, a `UserApplication` has multiple `SysNode` parents.
- `UserApplication` objects have one child each for each `SysNode` on which they may run. That child is usually an `andOP` object type and must have its `HostName` attribute set to the `SysNode` name to which it refers.

  By default, the configuration wizards generate a name of the form <*application_name*>`_Host_`<*hostname*> for each of these `andOP` objects.
- Each `SysNode` and `UserApplication` object can appear only once in the graph.
- Every instance of an object can only be used once in a configuration.
- Objects that belong to different `UserApplication` object types cannot depend on each other.
- A leaf object must always have a detector.
- There must not be any circular dependencies. For example, if A depends on B, and B depends on C, then C cannot depend on A.

6.1.2 Resource description

The configuration wizards generate descriptions for each application's required resources. The descriptions include the following:

- What action occurs if the state of a resource changes
- How RMS should configure or de-configure a resource
- What interdependencies exist between the resources

The configuration file uses a typical RMS meta-language and has the following characteristics:

- Objects represent resources.
- 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.

Upon startup, RMS interprets the configuration file and distributes the information to all cluster nodes.

6.1.3 Messages

In RMS, objects exchange messages with the following:

- Detectors
- Command interface
- GUI
- Other objects

Objects exchange this data 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 `userApplication` object and then forwards the requests to the children. The `userApplication` can also generate its own requests on the basis of changes to its state (such as a change over to the `Faulted` state). Requests always emanate from the `userApplication` and are forwarded from the parent to the child (top-down).

The processing of state change messages between `Offline` and `Online` differs as follows:

- State change to `Offline`—Offline processing is top-down; for example, a mount point is unmounted first, and then the underlying device is deconfigured.
- State change to `Online`—Online processing is bottom-up. While the online request travels down the tree from the `userApplication` to the leaf object(s), RMS executes the actual state change bottom-up; for example, first RMS configures the mirror, than it mounts the file system on the mirror.

6.1.4 State transition rules

RMS uses state transition rules to define which messages (requests or state changes) trigger what reaction in what situation. The fundamental concept is clarified in the following description of RMS procedures.

6.2 States and scripts

The chapter “Introduction” on page 9 introduced the concepts of scripts and the functions they perform. Scripts are divided as follows:

- Request-triggered scripts—Designed to produce a change in a state
- State-triggered scripts—Represent a reaction to a specific state

Request-triggered scripts are as follows:

- `InitScript`
- `PreOnlineScript`
- `PreOfflineScript`
- `PreCheckScript`
- `OnlineScript`
- `OfflineScript`

State-triggered scripts are as follows:

- `PostOnlineScript`
- `PostOfflineScript`
- `OfflineDoneScript`
- `FaultScript`
- `WarningScript`
- `StateChangeScript`

Post-online and post-offline scripts are generally state-triggered scripts. For example, if an online script executes successfully, RMS invokes the `PostOnlineScript` when the resource goes online. A similar situation is applicable for the `PostOfflineScript`.

Scripts are attributes of nodes. The use of scripts is always optional. The base monitor interprets unused script attributes (except the `ShutdownScript`) as scripts that terminate immediately and successfully (that is, as a script that contains only the line `exit 0`). If the `ShutdownScript` is not defined, then it is ignored. All script types can be used with all nodes, except for `SysNode`s, for which only a `FaultScript` and a `ShutdownScript` can be defined. Any changeover of a `SysNode` to the `Online` and `Offline` states is not subject to the control of RMS.

6.3 Initializing

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

The following is necessary information for identifying the state:

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

Two conclusions can be drawn from the above:

- Leaf nodes 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 node to the `userApplication`. Every node above the leaf node first requires the state of its children before it is able to determine its own state.

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

The initializing processes of `userApplication` nodes are independent of each other. Therefore, one application can be initialized, whereas another application can be `Unknown`.

The initializing process of `SysNode`s is also independent. Initially in the `Unknown` state, a `SysNode` exits after receiving the detector report. Thus, it does not wait for messages from its children (`userApplication`). This again illustrates the independence of the parent/child relationship between the `userApplication` and the `SysNode`.

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

6.4 Online processing

The online processing for a `userApplication` is independent from the online processing of another `userApplication`. Normally, this process results in the `userApplication` transitioning to the `Online` state. The following situations can prevent the `userApplication` from transitioning to the `Online` state:

- `PreCheckScript` determines that the `userAppliction` should not come online.
- Fault occurs during online processing.

These situations are discussed in detail in later sections.

6.4.1 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 165).

The following actions can generate an online request:

- Manual request using the GUI
- Manual request using the CLI
- Automatic request at RMS startup
- Automatic request when a fault occurs

6.4.1.1 Manual methods

Both 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 order of the children in the `userApplication` node determines the `SysNode` priorities.
- Directed switch—The user selects the `SysNode`. The `userApplication` is switched to a specific `SysNode`.

In both priority and directed switches, only `SysNode`s 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.

6.4.1.2 Automatic methods

Both 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 host.

Automatic switch at RMS startup only occurs under the following conditions:

- All `SysNode`s associated with a specific application are online.
- `userApplication` is not online on any other cluster node.
- `AutoStartUp` attribute of the `userApplication` is enabled.

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

Automatic request when a fault occurs

RMS initiates a priority switchover when it detects either a fault of a `userAp-plication` or a fault of a `SysNode` on which a `userAppliction` was online. This automatic switchover occurs only if the `AutoSwitchOver` attribute of the `userApplication` is enabled.

6.4.2 Online processing in a logical graph of a userApplication

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 node changes to the `Wait` state, but no script is initiated.
3. Child receives the request. The pre-online script is initiated in the leaf nodes.
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` node is the final node to execute its pre-online script; it then generates an online request, which is passed to the leaf nodes. 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 node detector signals the `Online` state. If a node 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 node exits the `Wait` state and changes to the `Online` state.

As shown previously, leaf nodes in an RMS configuration require at least an `OnlineScript`. The scripts of the leaf nodes execute first during online processing. The system then waits until the node changes to the `Online` state. On the other hand, nodes with children do not need an online script if they can be brought online in the `OnlineScript` of a child.

i Resource nodes that cannot go offline due to physical reasons (such as physical disks) are an exception to the rule that leaf nodes require online scripts. These nodes are identified in RMS configurations with the attribute `LieOffline=1`; (refer also to the section “Node does not have an Offline state” on page 159).

In 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 either not controlled by RMS at all, or it is started in the online script (and possibly in the post-online script) of the `userApplication`.

Even in the latter case, the `userApplication` is online only means that this script has been completed successfully. Whether and to what extent this fact permits statements to be made as to the state of the application is decided exclusively in the application, and cannot be influenced by RMS.

6.4.3 PreCheckScript

Before online processing begins, the `PreCheckScript` determines if online processing is needed or even possible. This procedure is required since some applications may be unable to start during online processing, thus causing the application to become `Faulted`.

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` node transits into the `Wait` state. If the `PreCheckScript` fails, the `userApplication` node transits back into its previous state, usually `Offline` or `Faulted`.

AutoSwitchOver

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

6.4.4 Fault situations during online processing

If an error situation occurs during online processing, the affected node commences fault processing and notifies its parent of the error (see also the section “Fault processing” on page 159). The following can cause faults during online processing:

- Detector signals the `Faulted` state.
- Detector signals the `Offline` state for a node that was reported as `Online`.

- Script fails with an exit status other than `0`.
- Script fails with a timeout.
- Detector does not detect the node as online within a specific period after the `OnlineScript` completes.
- Child of an `AND` node indicates a `Fault`.
- All children of an `OR` node signal `Fault`.

6.4.5 userApplication 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 nodes 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 nodes as usual. The only difference from the section "Online processing" on page 152 is that any nodes that are already online forward the request or the responses without executing their scripts and without changing to the `Wait` state.

A typical example of a node which is always online when RMS is initialized is a node for a physical disk (node type: `disk`) since physical disks cannot be deconfigured.

i Due to the property of the PXRE, the physical disk can be deconfigured on Solaris.

No request while online

If a `userApplication` does not receive an online request when it is already online and RMS is initialized, no explicit online processing is carried out in the logical graph. The `userApplication`, however, notifies its `Online` state to the other RMS monitors on the other hosts in the cluster to ensure that no corresponding application goes online on one of these hosts. A primary objective of RMS is to ensure that no data losses occur as a result of simultaneous activity of an application on several hosts.

It can be extremely damaging if a `userApplication` is online on more than one host directly 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.

6.5 Offline processing

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

6.5.1 Offline request

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 159). This offline request is also a primary precondition for any subsequent switchover.

6.5.2 Offline processing in a logical graph of a userApplication

Unlike online processing, the direction of offline processing is from the `userAp-plication` to the leaf node (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 script, each child node changes to the `Wait` state, executes its pre-offline script, and forwards the request.
4. As soon as the leaf nodes have completed their pre-offline script, they send a corresponding message (confirmation of successful 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 node without a detector).
7. The `userApplication` then generates the actual offline request.

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

- The offline script is executed first.
- The post-offline script is started after the detector report `Offline` has arrived.
- The request is forwarded to the children after the post-offline script has completed.

As illustrated, the `userApplication` is the final node to go offline. After the offline process completes, the `userApplication` notifies the corresponding `userApplication` nodes on the other hosts that the application has gone offline.

In the case that the `hvshut` command is used, RMS initiates offline processing, and the `userApplication` checks the state of other `userApplication` nodes on the local host. RMS is then terminated if all of these local `userApplication` nodes are offline.

6.5.3 Fault situations during offline processing

The section “Fault processing” on page 159 describes the processing of any faults that occur during offline processing. The following can cause faults during offline processing:

- Detector indicates the `Faulted` state.
- Detector signals the `Online` state for a node that was reported as `Offline`.
- Script fails with an exit status other than `0`.
- Script fails with a timeout.
- Node is not detected by the detector as being `Offline` within a specific period after the offline script completes.
- Child of a node indicates a fault.

6.5.4 Node is already offline

If a node is already offline at the start of offline processing (a situation which can occur only in nodes below an `OR` node), the request is merely passed through (similar to the situation in online processing); scripts are not executed and the `Wait` state is not entered.

6.5.5 Node does not have an Offline state

RMS covers an extremely wide range of system conditions, including monitoring resources that have no `Offline` state. Physical disks are an example of such nodes because they are monitored but cannot be deconfigured. For this purpose, RMS provides the attribute `LieOffline` to indicate that the resource has no `Offline` state. This attribute is set by default for physical disks (node type: `disk`) and does not have to be explicitly specified.

During offline processing, a node identified with `LieOffline` reacts in the same way as any other node and, in particular, when all pre-, post- and offline scripts are run. The reaction of the node with respect to its parent is also the same as if the node had been successfully deconfigured; that is, it “lies.” A node with `LieOffline` set does not wait for an offline report of the detector after the offline script has executed; 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.

6.6 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.

6.6.1 Faults in the online state or request processing

When a detector indicates a fault for an online node whose corresponding `userApplication` is also online, RMS executes the fault script of the node. An equivalent fault condition occurs if the detector indicates that a previously online node 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 `OR` nodes. These react to the fault message only if no other child is online. If another child of the parent is online, RMS terminates the fault processing at this point.

If there is no intermediate `OR` node that intercepts the fault message, it reaches the `userApplication`. The `userApplication` then executes its fault script. There are four subsequent cases possible during processing. These attributes are set for the `userApplication` in the configuration file according to the needs of the application.

These fault processing combinations are as follows:

- `AutoSwitchOver` is set
- `PreserveState` is set but `AutoSwitchOver` is not set
- Neither `AutoSwitchOver` nor `PreserveState` are set
- Both `AutoSwitchOver` and `PreserveState` are set

AutoSwitchOver only

If the `AutoSwitchOver` attribute is set for the `userApplication`, the process is as follows:

1. The `userApplication` attempts to initiate the switchover procedure. For this purpose, the application on the local host 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 host (see the section “Switch processing” on page 165). However, the `userApplication` is now in the `Faulted` state—unlike the situation with a normal offline request. This prevents the possibility of an application returning to the host in the event of another switchover.

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

i A failure during offline processing is called a double fault. A double fault causes the machine to be eliminated if the `userApplication` halt flag is set.

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 host is shut down immediately if RMS cannot resolve a double-fault state. The other hosts detect this as a system failure, and RMS transfers the applications running on the failed host to another host.

PreserveState without AutoSwitchOver

If the PreserveState attribute is set and the `AutoSwitchOver` attribute is not set in the `userApplication`, the process is as follows:

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

Use this attribute if an application can remedy faults in required resources.

Neither AutoSwitchOver nor PreserveState

If neither the `AutoSwitchOver` attribute nor the `PreserveState` attribute is set, 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).

Both AutoSwitchOver and PreserveState

If both the `AutoSwitchOver` attribute and the `PreserveState` attribute are set, RMS ignores the `PreserveState` attribute and responds as if only the `AutoSwitchOver` attribute were set.

Directed switch fault

A special case occurs when a directed 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 not set. Switchover had evidently been requested at this time by the system administrator who sent the directed switch request online. The target host of the switchover procedure may not be the host with the highest priority; it is the host explicitly specified in the directed switch request.

6.6.2 Offline faults

Even if a `userApplication` is not online on a host, RMS still monitors the nodes configured in the graph of the `userApplication`. If a detector indicates a fault in such a node, 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 `AND` node could be offline, although one of its children is `Faulted`.

RMS contains this design on the basis that the mandatory dependency correlation between the nodes in a `userApplication` graph exist only if the `userApplication` is to run.

In the offline case, RMS treats the nodes as individual instances and does not evaluate their mutual interdependencies. However, an exception occurs when the `ClusterExclusive` attribute is set. If a `userApplication` is offline, has the `ClusterExclusive` attribute set, and has children that are not offline, then `hvdisp` will display the state `Inconsistent` instead of `Offline` for this `userApplication`.

6.6.3 AutoRecover attribute

A node of the type `mount` is one example of a node that can enter a `Faulted` state due to reasons that are easily and automatically remedied. A fault that occurs in the node 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, programmers can configure the `AutoRecover` attribute in RMS. If a fault then occurs when the `userApplication` is online, the online script is invoked before the fault script. If the node enters the `Online` state again within a specific period after the online script has been executed, the node goes online again, and fault processing does not take place.

RMS only evaluates the `AutoRecover` attribute when the node 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 nodes with a detector. The `AutoRecover` attribute is not relevant even if a fault occurs during request processing or in the `Offline` state.

i The `AutoRecover` attribute in RMS is not set as a default for any node type. The specialist who configures RMS must decide whether to use the attribute.

6.6.4 Fault clearing

After successful fault processing, the resource nodes will be offline, and the `userApplication` will be faulted. If offline processing fails as a result of the fault, or if the `PreserveState` attribute were used, at least part of the graph will be in a `Wait` state.

In all of the above states, the `userApplication` blocks the normal requests (such as a switch request), since the base monitor assumes that at least some of the resources are not available. RMS can only resume normal operation after the system administrator has remedied the cause of the fault. The following options are available for notifying the base monitor that the cause of the fault has been cleared (fault clearing):

1. After clearing the fault condition, the system administrator can use the following command to send a clear-fault request to the `userApplication`:

   `hvutil -c` *<userApplication>*

   This then starts further offline processing. If the fault has cleared, the entire tree will be offline. If required, the system administrator can reset the `userApplication` to the `Online` state with a switch request.

   i Invoking `hvutil -c` results in further online processing if the fault occurs below an `orOp` node. In such cases, the node and its parents up to the `OR` node are faulted. However, the fault has not been forwarded to the `userApplication`. The `userApplication` will thus still be online.

2. The system administrator can use the following command to make a forced-online request:

   `hvswitch -f` *<userApplication>* *< target_host>*

The `userApplication` starts online processing and, assuming that the fault is cleared, resets the application to the `Online` state.

> i A forced online request will fail if fault processing has failed or if the `PreserveState` attribute was set. In these cases, it is likely that individual nodes will be in an undefined `Wait` state in which RMS cannot process an online request to ensure consistency.

Forced-online request

A forced-online request can be sent to a target host that is not the host on which the application was running when the fault occurred. This would be an instance of a forced switchover. Again, the forced request will not be successful if fault processing failed on the previous host (a forced switchover does not automatically mean a forced-offline request on the previous host).

If fault processing succeeded on the previous host, the forced-online request has the same effect as a local forced-online request sent to the target host. Online processing is initiated, even if individual nodes are faulted.

6.6.5 SysNode faults

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

- `SysNode` detector loses contact with RMS.
- `SysNode` detector loses contact with a cluster host.
- `LEFTCLUSTER` event occurs.

When any of these events happen, RMS must first ensure that the host with which contact was lost is 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 host is verified by the SF, all `userApplication` nodes that were `Online` on the affected cluster host are priority switched to surviving cluster hosts.

Descriptions of the shutdown methods are provided in the sections that follow.

6.6.5.1 Operator intervention

If SF fails, then operator intervention is required. The indication that operator intervention is required is the persistent `Wait` state of any `SysNode` in the cluster. In this instance, a persistent `Wait` state is defined as a `SysNode Wait` state that lasts longer than the SCON reply time added to the script timeout for the `ShutdownScript`.

The value of the SCON reply time can be found by executing the following:

```
/opt/SMAW/SMAWRrms/bin/hvenv | grep HV_SCON_REPLY_TIME
```

The value of the script timeout for the `ShutdownScript` can be found by executing the following:

```
/opt/SMAW/SMAWRrms/bin/hvdisp SysNode_name | grep Script-
Timeout
```

Alternately, the administrator can look for a message in the switchlog indicating that operator intervention is required.

After determining that operator intervention is required, the operator must perform the following:

1. Manually shut down the cluster host indicated by the `SysNode` in the `Wait` state.
2. Issue the `hvutil -u` *SysNode_name* command on a surviving cluster host.

6.7 Switch processing

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

6.7.1 Switch request

Switch requests are divided as follows:

- Priority switch request—RMS identifies the target host according to the host priority as defined in the configuration (see the description of the `PrioityList` attribute in the chapter “Appendix—Attributes” on page 325).
- Directed switch request—The user specifies the target host.

The types of switches are divided as follows:

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

In switch processing, RMS performs the activities in Table 6 depending on the switch scenario.

Activity	**Switch-over**	**Switch-online**
The `userApplication` generates a switch request when RMS starts. This occurs if the `AutoStartUp` or `AutoSwitchOver` attributes are set (only priority requests).	X	X
The system administrator generates a switch request by means of the command interface (priority as well as directed switch request are both possible).	X	X
i For priority request, the configured priorities of the hosts relative to the affected applications determines the target host.		
RMS forwards the request to the host on which the `userApplication` node is currently online.	X	
RMS forwards the request to the host on which `userApplication` is to go online.		X
To establish whether its local graph contains a fault condition which would prevent the application from going online, the `userApplication` communicates with its complementary node on the target host. RMS functions as follows if such a fault condition exists: Terminates ● switch processing (directed switch) ● Identifies the next host in priority as the new target host (priority switch). If no new target host is identified, RMS terminates switch processing.	X	

Table 6: Switch processing activities

Activity	**Switch-over**	**Switch-online**
The `userApplication` carries out local offline processing, stops, and thus deconfigures the ongoing application.	X	
The `userApplication` transmits the online request to the corresponding node on the target host.	X	
The `userApplication` on the target host carries out local online processing.	X	X

Table 6: Switch processing activities

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

6.7.2 Extreme situations during switch processing

In rare cases, fatal fault situations of varying severity can occur during switch processing; for example, the relevant host can crash or communication between the hosts can (temporarily) fail. RMS resolves these situations by means of a complex scenario based on timeout-handling recovery measures and by recalculation. These measures are carried out transparently to the user.

It is important to realize that under extreme circumstances, inconsistencies that RMS cannot resolve can occur in the cluster. To minimize the damage in the case of data resources that are available for parallel access, RMS blocks any further requests by entering a cluster-wide loop state. When the cause of the problem has been identified and cleared, the system administrator must stop and restart RMS for the entire cluster. This guarantees consistency by reinitializing the internal RMS states.

i Stopping and restarting RMS does not mean that all applications under the control of RMS have to be stopped. RMS provides a command (`hvshut -L`) that enables system administrators to stop RMS without performing offline processing for the applications. The system administrator can then restart RMS while the applications are running (see also the section “Online processing” on page 152).

7 Troubleshooting

This chapter discusses some PRIMECLUSTER facilities for debugging the RMS product from both the command line interface (CLI) and from the Cluster Admin graphical user interface (GUI). This chapter provides details on log files, their location, how to turn on logging levels, how to view logs from the GUI, and how to view log files from CLI.

This chapter discusses the following:

- The section “Overview” on page 169 summarizes the troubleshooting process.
- The section “Debug and error messages” on page 171 describes RMS debug and error messages.
- The section “Log files” on page 172 identifies and explains the RMS log files.
- The section “Using the log viewer” on page 174 explains the log viewer facilities.
- The section “Specifying the log level” on page 182 specifies and explains the log levels.
- The section “Interpreting log files” on page 185 explains the meaning of the data in the log files.
- The section “System log” on page 186 describes the system log.
- The section “Wizard log files” on page 187 details the RMS Wizard log files.
- The section “PCS log files” on page 191 lists the locations of the PCS log files.
- The section “RMS troubleshooting” on page 191 supplies solutions to problems that could occur while using RMS.

7.1 Overview

The RMS troubleshooting process usually begins after you observe an error condition or state change in Cluster Admin in one of the following areas:

- Clusterwide table
- RMS tree

- Graph

The clusterwide table contains summary information and is a good place to start looking for error conditions. For additional details, you can look at the RMS tree or the graph. Depending on whether you need to look at the switchlogs or application logs, you can then use the log viewer facility to view the log files.

The log viewer has search facilities based on the following:

- Keywords
- Severity
- Non-zero exit codes

Search for causes of errors using the keywords and the date range fields. For emergency, alert, and critical conditions, you can do a search based on severity. For proactive troubleshooting, you can perform a search based on severity for the `error`, `warning`, `notice`, and `info` severity codes.

i It is recommended that you periodically use the log viewer and check the log files based on the severity levels to avoid serious problems. If you cannot diagnose the cause of a problem, look at the log viewer from two or more nodes in the cluster.

i Refer to the section “RMS troubleshooting” on page 191 for an explanation on corrective action.

Resolve error conditions as follows:

1. Use the Cluster Admin GUI.
2. View the log files if needed.
3. Change log levels to get more details.
4. If you cannot resolve an error condition with the GUI, you can use the command line interface. Use standard UNIX commands.
5. If a problem persists, check if it is a non-RMS issue and refer to the appropriate manual.
6. Check for system-related issues like operating system, hardware, or network errors.
7. Contact field support if you cannot resolve the issue.

7.2 Debug and error messages

RMS writes debug and error messages to log files when its components (such as the base monitor or detectors) operate. The default setting is for RMS to store these files in the `/var/opt/SMAWRrms/log` directory. Users can change the directory with the `RELIANT_LOG_PATH` environment variable, which is set in the `hvenv.local` file.

When RMS starts, logging begins. The default setting is for the base monitor to write all error messages to its log file or to `stderr`. Normally, you do not need to change the default setting because the default options allow for very detailed control of debug output.

If required, you can use the base monitor to record every state and message of any node. However, in most cases, the information requires a detailed knowledge of internal RMS operation to interpret the debug output, which can only be evaluated by service personnel.

For the administrator of an RMS cluster, evaluating the `switchlog` file is normally sufficient. This file records all important RMS actions; for example, incoming switch requests or faults that occur in nodes.

i There are also configuration-specific log files in the log directory. It is recommended that administrators evaluate these if necessary. The names of these log files depend on the configuration that was set up using the configuration wizards (RMS Wizard Tools or PCS). Consult the RMS Wizard Tools or PCS online documentation for further information.

The following log files can also be used for problem solving:

- `hvdet_nodelog`
- `bmlog`

7.3 Log files

Table 7 identifies and explains the RMS log files contained in `/var/opt/SMAWRrms/log`.

Module	**File Name**	**Contents**
base monitor	`tracelog`	Records all messages between objects and all modification instructions. The default is off.
base monitor	`abortstartlog`	This file contains records about bm exit conditions to assist support personnel in determining why RMS failed to start. This file is generated if the following message appears during startup: `FATAL ERROR: RMS has failed to start!`
base monitor	`bmlog`	General RMS error and message logging information ranges from simple message reporting to more complete information. The error log level determines the contents of this file, which is specified when the base monitor is started. Refer to the section “Specifying the log level” on page 182 for more information. Includes all messages received by the base monitor at runtime. Limited use to administrators since turning on log level flags consumes a great deal of disk space. By default, RMS places no messages in `bmlog`.

Table 7: Log files

Module	**File Name**	**Contents**
Everything (base monitor, generic detector, node detector)	`switchlog`	Operational events, such as resource switches or bugs. Normally, `switchlog` is the only log file users need to examine.
generic detector	*<program>*`log`	All messages and job assignments received by the detector. Also contains resource state change information and all error messages. *program* is the name of the detector in the *<RELIANT_PATH>* directory.
node detector (`hvdet_node`)	`hvdet_nodelog`	Messages from the built-in node detector, `hvdet_node`.

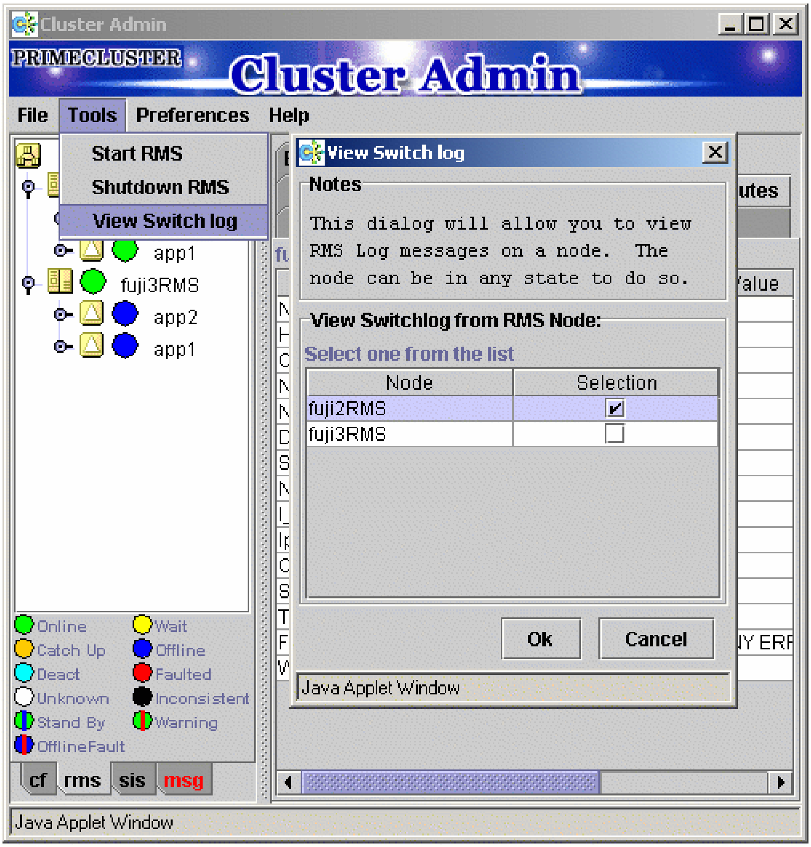
Table 7: Log files

7.4 Using the log viewer

Invoke the log viewer for the RMS `switchlog` file as follows:

1. Right-click on a `SysNode` in the RMS Tree.
2. Select *View switch log.*

Figure 114 shows how to invoke the log viewer.



This dialog will allow you to view RMS Log messages on a node. The node can be in any state to do so.

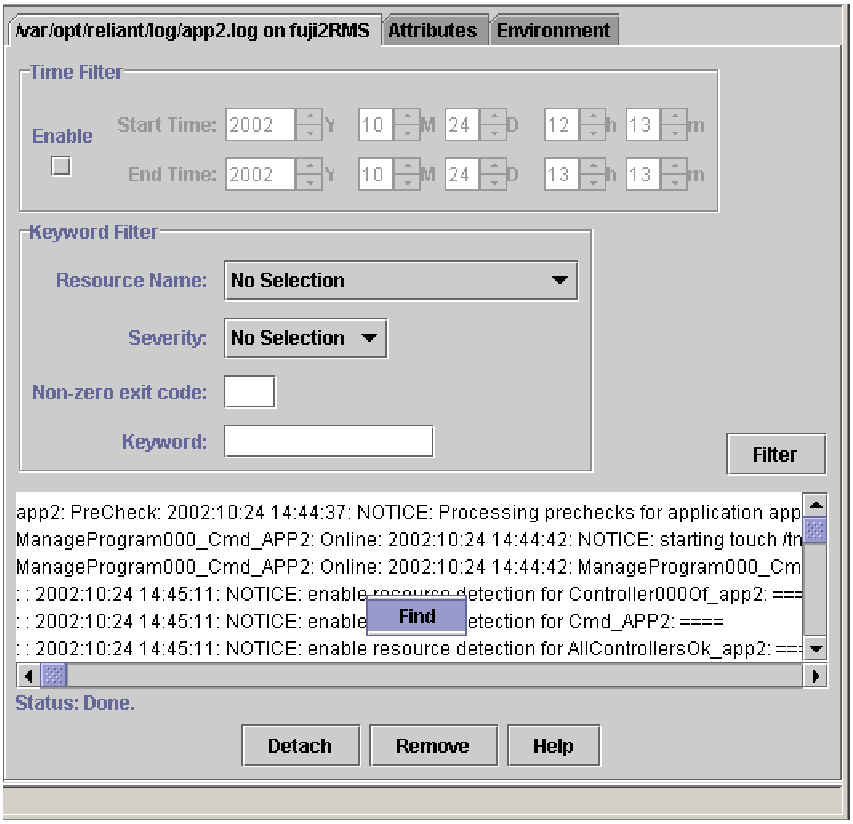
Node	Selection
fuji2RMS	☑
fuji3RMS	☐

Figure 114: Invoking the log viewer

You can search the logs based on any of the following:

- Resource name
- Date/time range
- Keyword filter
- Severity levels
- Exit codes

You can also search in the log display window by right-clicking on the displayed text. This brings up a *Find* pop-up window (Figure 115).

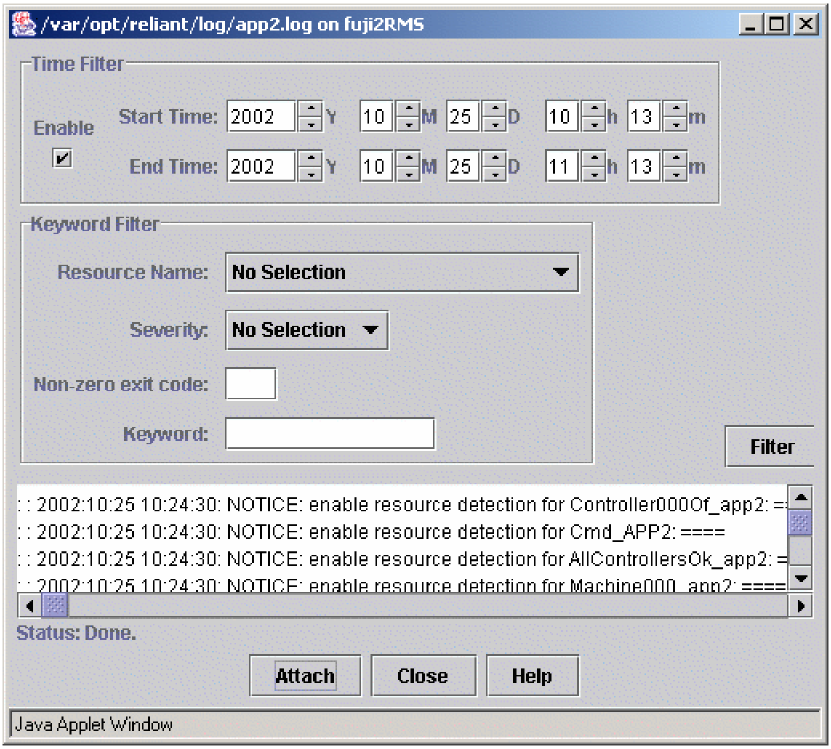

/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Environment
Time Filter
Enable
Start Time: 2002 Y 10 M 24 D 12 h 13 m
End Time: 2002 Y 10 M 24 D 13 h 13 m
Keyword Filter
Resource Name: No Selection
Severity: No Selection
Non-zero exit code:
Keyword:
Filter
Find
Status: Done.
Detach
Remove
Help

```
app2: PreCheck: 2002:10:24 14:44:37: NOTICE: Processing prechecks for application app
ManageProgram000_Cmd_APP2: Online: 2002:10:24 14:44:42: NOTICE: starting touch /tn
ManageProgram000_Cmd_APP2: Online: 2002:10:24 14:44:42: ManageProgram000_Cm
: : 2002:10:24 14:45:11: NOTICE: enable resource detection for Controller000Of_app2: ===
: : 2002:10:24 14:45:11: NOTICE: enable      etection for Cmd_APP2: ====
: : 2002:10:24 14:45:11: NOTICE: enable resource detection for AllControllersOk_app2: ===
```

Figure 115: Find pop-up window

Detach the log by clicking on the *Detach* button. Use the *Attach* button to attach it again.

Figure 116 shows a detached log.



Start Time: 2002 Y 10 M 25 D 10 h 13 m
End Time: 2002 Y 10 M 25 D 11 h 13 m

Resource Name: No Selection
Severity: No Selection
Non-zero exit code:
Keyword:

```
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Controller000Of_app2: =
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Cmd_APP2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for AllControllersOk_app2: =
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine000_app2: ====
```

Status: Done.

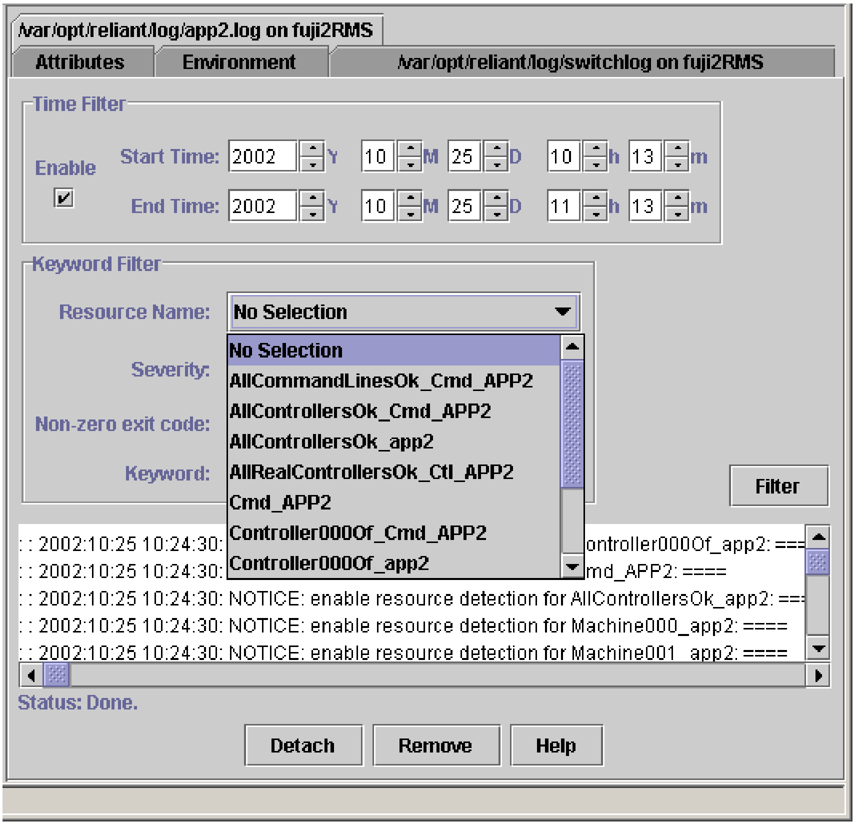
Figure 116: Detached log

7.4.1 Search based on resource

Searches based on the name of the resource apply only to application logs. Search the log files based on the name of a resource as follows:

1. Select the name of the resource from the pull-down list.
2. Press the *Filter* button.

Figure 117 shows the window for a search based on the resource name.


/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Environment
/var/opt/reliant/log/switchlog on fuji2RMS
Time Filter
Enable
Keyword Filter
Filter
Detach
Remove
Help

Start Time: 2002 Y 10 M 25 D 10 h 13 m
End Time: 2002 Y 10 M 25 D 11 h 13 m

Resource Name: No Selection

Severity:

Non-zero exit code:

Keyword:

No Selection
AllCommandLinesOk_Cmd_APP2
AllControllersOk_Cmd_APP2
AllControllersOk_app2
AllRealControllersOk_Ctl_APP2
Cmd_APP2
Controller000Of_Cmd_APP2
Controller000Of_app2

```
: : 2002:10:25 10:24:30:                              ontroller000Of_app2: ===
: : 2002:10:25 10:24:30:                              md_APP2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for AllControllersOk_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine000_app2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine001_app2: ====
```

Status: Done.

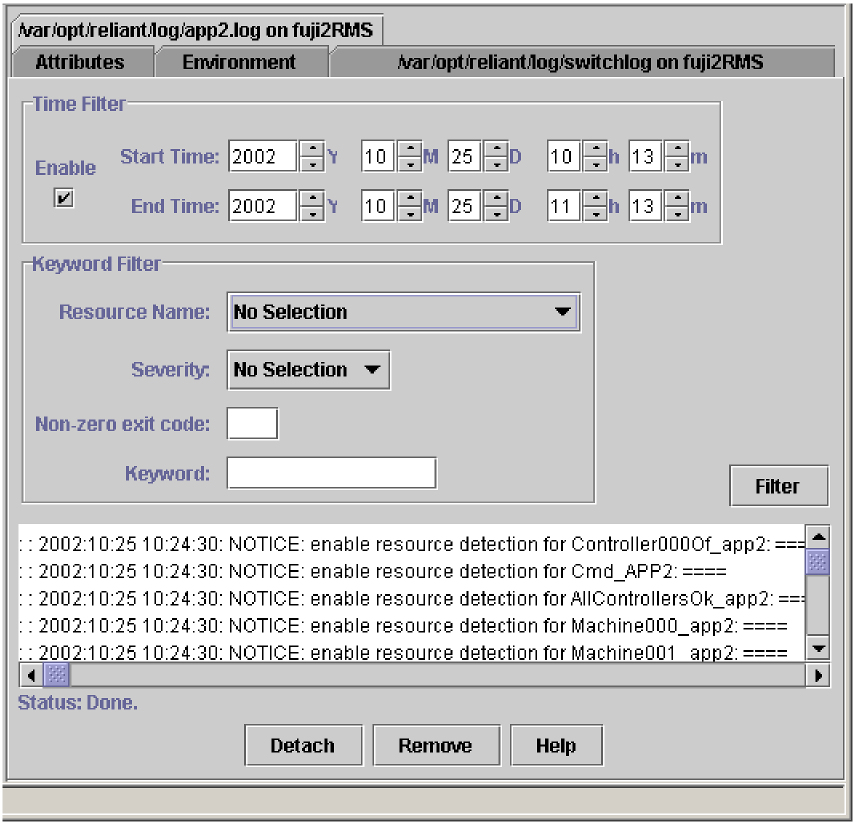
Figure 117: Resource-based search

7.4.2 Search based on time

Search the log files based on the date and time range as follows:

1. Specify the start and end times for the search range.
2. Click on *Enable*.
3. Press the *Filter* button.

Figure 118 shows the results for a search based on the time filter.


/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Environment
/var/opt/reliant/log/switchlog on fuji2RMS
Time Filter
Keyword Filter
Filter
Detach
Remove
Help

Enable ✓

Start Time: 2002 Y 10 M 25 D 10 h 13 m

End Time: 2002 Y 10 M 25 D 11 h 13 m

Resource Name: No Selection

Severity: No Selection

Non-zero exit code:

Keyword:

```
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Controller000f_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Cmd_APP2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for AllControllersOk_app2: ==
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine000_app2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine001_app2: ====
```

Status: Done.

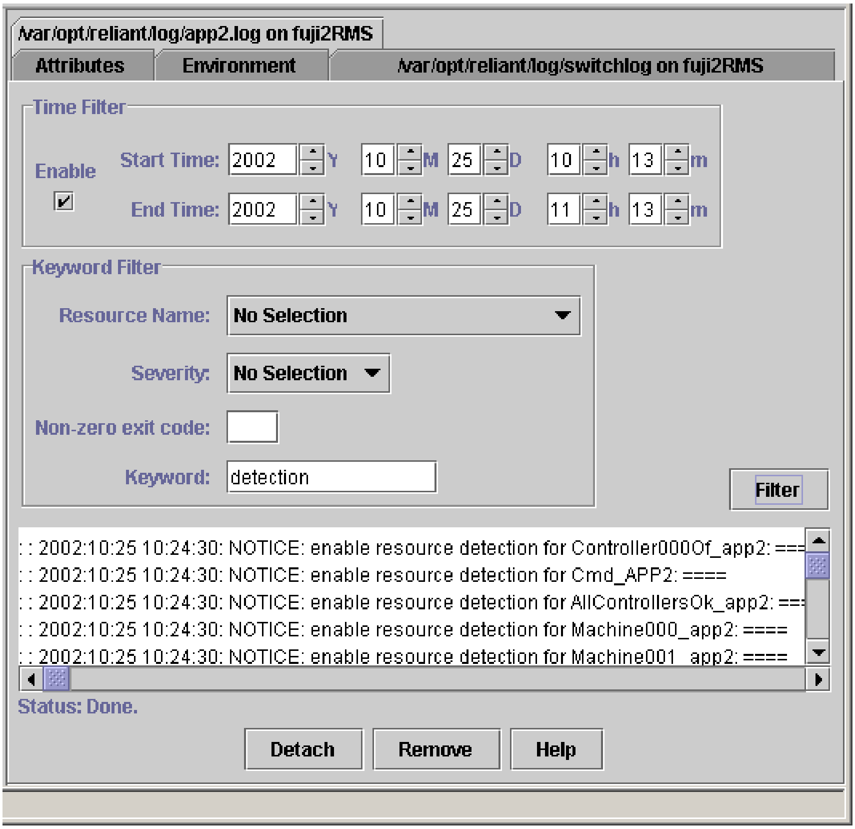
Figure 118: Results of time-based search

7.4.3 Search based on keyword

Search the log files based on a keyword as follows:

1. Enter a keyword.
2. Click on the *Filter* button.

Figure 119 shows an example of a log file search based on a keyword.


/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Environment
/var/opt/reliant/log/switchlog on fuji2RMS
Time Filter
Keyword Filter
Filter
Detach
Remove
Help

Enable ✓

Start Time: 2002 Y 10 M 25 D 10 h 13 m

End Time: 2002 Y 10 M 25 D 11 h 13 m

Resource Name: No Selection

Severity: No Selection

Non-zero exit code:

Keyword: detection

```
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Controller000Of_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Cmd_APP2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for AllControllersOk_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine000_app2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine001_app2: ====
```

Status: Done.

Figure 119: Results of keyword-based search

7.4.4 Search based on severity levels

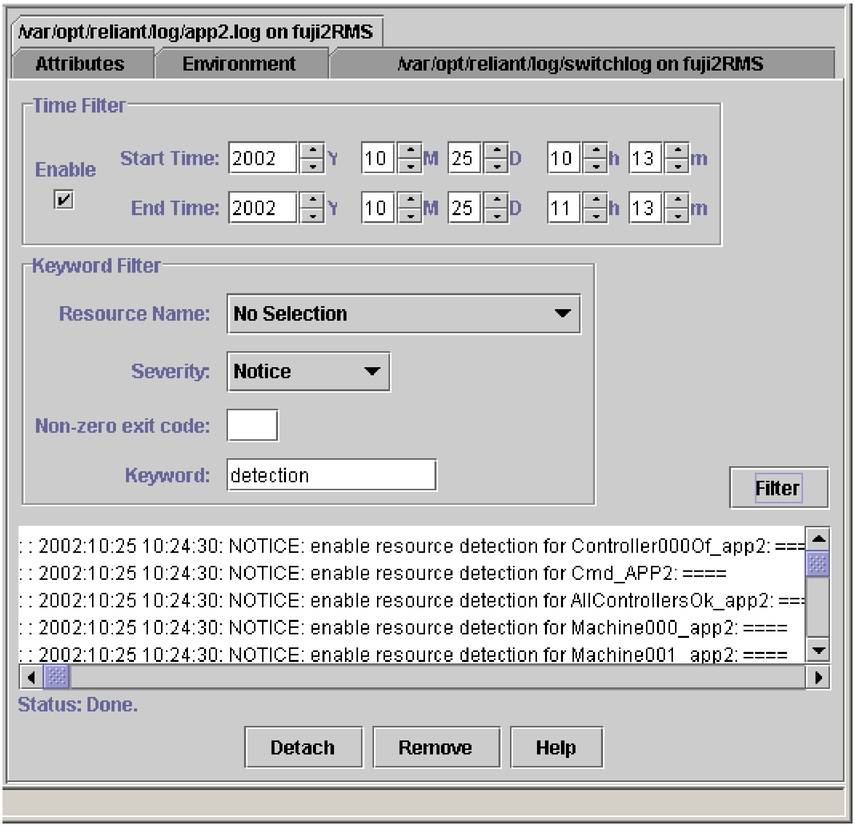
Search the log files based on severity levels as follows:

1. Click on the *Severity* button.
2. Choose one of the severity levels as described in Table 8.
3. Click on the *Filter* button.

Severity level	Description
Emergency	Systems cannot be used
Alert	Immediate action is necessary
Critical	Critical condition
Error	Error condition
Warning	Warning condition
Notice	Normal but important condition
Info	For information
Debug	Debug messages

Table 8: Descriptions of severity levels

Figure 120 is an example of a log file search based on a severity level.


/var/opt/reliant/log/app2.log on fuji2RMS
Attributes
Environment
/var/opt/reliant/log/switchlog on fuji2RMS
Time Filter
Keyword Filter
Filter
Detach
Remove
Help

Enable ✓

Start Time: 2002 Y 10 M 25 D 10 h 13 m

End Time: 2002 Y 10 M 25 D 11 h 13 m

Resource Name: No Selection

Severity: Notice

Non-zero exit code:

Keyword: detection

```
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Controller000Of_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Cmd_APP2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for AllControllersOk_app2: ===
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine000_app2: ====
: : 2002:10:25 10:24:30: NOTICE: enable resource detection for Machine001_app2: ====
```

Status: Done.

Figure 120: Results of severity-level-based search

7.5 Using the hvdump command

The `hvdump` command is used to get debugging information about RMS on the local node. Independent of the base monitor running on the local node, invoking `hvdump` causes it to gather PRIMECLUSTER product and system files that will be used to diagnose the problem. For a detailed list of the information gathered, consult the `hvdump`(1M) manual page.

7.6 Specifying the log level

For further debugging information, use the `-l` *level* option of the `hvcm` or `hvutil` commands to activate various logging procedures.

i Specify logging with the `-l` (lowercase “L”) flag.

To activate logging, shut down RMS and restart it with one of the log levels described in Table 9, or use the `hvutil` command to set the logging after RMS has been started. The log level specified with the `-l` option is a list with numbers or a range. Separate levels by means of commas or spaces in the list. If a space is used as a list separator, include the entire argument between the braces. A level range is defined as *n1-n2*. This includes all log levels from *n1* up to and including *n2*. The -*n2* range is the same as `1`-*n2*. The *n1* range defines all log levels above *n1*. The *n1* value must be greater than or equal to 1.

All log levels refer to internal functions of the base monitor and are only relevant for service personnel. In addition, executing RMS with several active log levels will affect system performance. If log level `0` is defined, all possible log levels are activated. Valid log levels are listed in Table 9.

Log Level	Meaning
`0`	Turn on all log levels
`1`	Unused
`2`	Turn on detector tracing
`3`	Unused
`4`	Turn on `mskx` tracing (stack tracing of the base monitor)
`5`	Error or warning message
`6`	Heartbeats
`7`	Base monitor level
`8`	Detector error
`9`	Administrative command message
`10`	Basic-type level
`11`	Dynamic reconfiguration contracting level

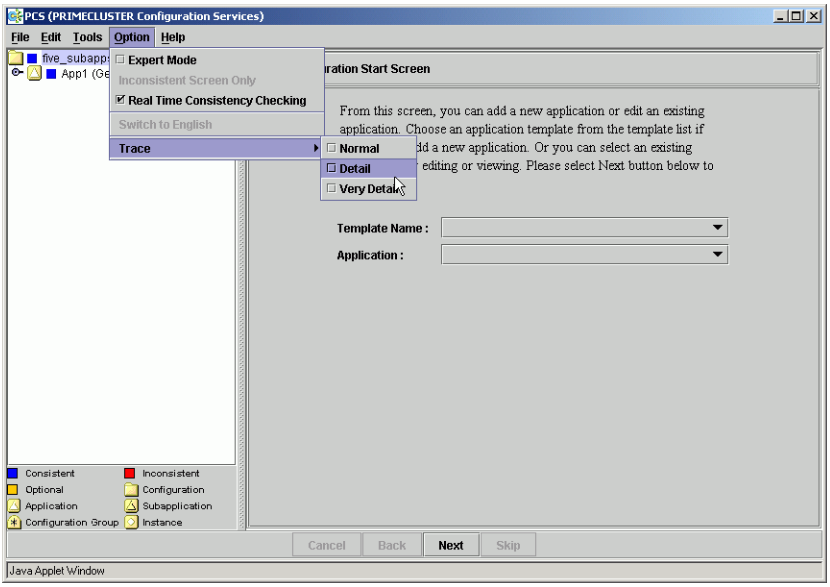
Table 9: Log levels

Log Level	Meaning
`12`	Unused
`13`	Token level
`14`	Detector message
`15`	Local queue level
`16`	Local queue level
`17`	Script level
`18`	`userApplication` contract level
`19`	Temporary debug traces
`20`	`SysNode` traces
`21`	Message level
`22`	bm tracelog

Table 9: Log levels

You can also control logging with the RMS Wizard Tools or PCS:

- From the Wizard Tools *Main configuration menu*, select *Configuration-Edit-Global-Settings –> DetectorDetails*. The menu that appears will allow you to set the log level for detectors in the configuration.
- From any PCS window, select the configuration (or any other item) in the left-hand tree, then use *Option –> Trace* and select the level of detail from the submenu (Figure 121).



From this screen, you can add a new application or edit an existing application. Choose an application template from the template list if dd a new application. Or you can select an existing editing or viewing. Please select Next button below to

Template Name :

Application :

Figure 121: Controlling the log level with PCS

7.7 Interpreting log files

Each process that makes up RMS generates three types of log messages: user, trace, and error. These log messages are contained in the following files:

`switchlog`	Records RMS events relevant to the user, such as switch requests and fault indications.
<program>`log`	Records trace messages or error messages for *program*. For example, messages from `bm`, the base monitor, are recorded in `bmlog`. The prefix for trace messages is as follows: *time*:*file*:*line*:. The prefix for error messages is as follows: *time*:*file*:*line*:`ERROR`

switchlog file

The `switchlog` file contains the following five message types:

- Informational messages (notices)
- Warning messages
- Error messages
- Fatal error messages
- Output from scripts run by RMS

The first four categories of messages all follow this format:

timestamp: (*error code*, *error number*): *message type*: *message*: *delimiter*

There is a colon-space (:) between each field of the message where the timestamp is defined as follows:

yyyy-*mm*-*dd hh*:*mm*:*ss*.*xxx*

Message type is defined as one of the following:

- `NOTICE`
- `WARNING`
- `ERROR`
- `FATAL ERROR`

Messages are any text generated by the RMS product. This text can contain one or more new lines. The delimiter is defined as a colon followed by a series of four equal signs (`:====`).

The last category of messages (output from scripts) follows no specific format and is merely the redirected standard output and standard error from all scripts defined within the RMS configuration file. For example:

```
2001-05-07 11:01:54.568: WARNING: InitScript does not exist.: ====
```

7.8 System log

The base monitor of RMS writes messages to the `switchlog` file and also writes the same messages to the system log. By default, all the RMS messages go to both the `switchlog` file and also to the system log.

`HV_SYSLOG_USE` is an environment variable that you can modify so that messages will or will not show on the system log. If you do not want the messages to go into the system log, then set `HV_SYSLOG_USE=0` in the `hvenv.local` file. Before changes can take effect, you must stop and restart RMS.

The default setting in `hvenv` is `HV_SYSLOG_USE=1`. This setting sends all RMS ERROR, FATAL ERROR, WARNING, and NOTICE messages to the system log and switchlog.

For Log3 RMS messages, the component number is 1080023.

hvlogcontrol

The `hvlogcontrol` utility prevents log files from becoming too large. Since large amounts of log files can take up disk space, `hvlogcontrol` limits the amount of log files to a specified amount set in one of the following environment variables selected by the system administrator:

- `HV_LOG_ACTION_THRESHOLD`
- `HV_LOG_WARN_THRESHOLD`
- `HV_SYSLOG_USE`

i `hvlogcontrol` is called automatically from the `crontab` file, so there is no manual page.

7.9 Wizard log files

The RMS Wizards log messages to files in the same log directory as is defined for RMS, according to the value set in the environment variable `RELIANT_LOG_PATH`. RMS Wizards logging can be broken down into two categories as follows:

- Messages resource detectors
- All other messages

Detector logging will be explained in more detail in section “RMS Wizards detector logging” on page 189.

Unlike RMS, which logs most of its messages in the `switchlog` file, the RMS Wizards log everything at an application level. All messages associated with a particular configured application are logged in the file *<RELIANT_LOG_PATH>*`/`*<application_name>*`.log`. The file is created when either offline or online processing for the application begins.

Each RMS Wizard process that is run, generates the following two types of log messages:

- User
- Debug

The log messages are contained in the following files:

- `switchlog`—Records RMS events relevant to the user such as switch requests and fault indications. The RMS Wizards record resource state transitions into the `switchlog` file.
- <*application_name*>`.log`—The application-specific log file records all messages associated with that application. The output from all scripts run by the application go into the log file.
- `hvdet_`*xxx*`.g`*nn*`log`—These are detector log files which record all relevant information regarding the resources they are monitoring, like all state transitions.

The format of most RMS Wizard messages is as follows:

resource_ name:*state*:*timestamp*:*message_type*:*Message*:*delimiter*

There is a colon (:) between each field of the message.

The *resource_name* field is the name of the particular resource node in the RMS graph whose script is running. This field may be empty if no resource is associated with the message.

The *state* field is an indication of the type of action that is being performed, and is the value as set by RMS in the environment variable `HV_SCRIPT_TYPE`. The field typically contains the values `online` or `offline`. The RMS Wizards also set the field with the value `PreCheck`, when a `PreCheck` script is being run. This field will be empty for messages of type `DEBUG` being printed.

The *timestamp* field contains the date when the message occurred and is written in the format *yyyy:mm:dd hh:mm:ss*, where *yyyy* is the 4 digit year; *mm* is the month number; *dd* is the day of the month; *hh* is the hour in the range of [0-23]; *mm* is the minute of the hour; *ss* is the number of seconds past the hour.

Message type is defined as one of the following:

- `DEBUG`
- `NOTICE`
- `WARNING`
- `ERROR`
- `FATAL ERROR`

Messages are any text generated by the RMS Wizard product. This text can contain one or more new lines. The delimiter is defined as a series of four equal signs (====).

Debug messages from scripts which are run can be forced by setting the environment variable `HV_SCRIPTS_DEBUG` to 1 in the `hvenv.local` file. The entry should appear as follows:

```
export HV_SCRIPTS_DEBUG=1
```

To turn off debug output, either remove the `HV_SCRIPTS_DEBUG` entry from the `hvenv.local` file, comment it out, or set the value to `0`.

When debugging problems, the `switchlog` file as well as the application-specific log file, and any appropriate detector log files may all need to be viewed and interpreted.

7.9.1 RMS Wizards detector logging

The RMS Wizard detectors log information to both the `switchlog` file and to their own detector log file `hvdet_`*xxx*`.g`*nn*`log` (for example, `hvdet_icmp.g64log`). All resource state changes are logged both to the `switchlog` file and to their own detector log file. Other detector messages are not logged to the `switchlog` file. A detector log file is created for each instance of a detector running.

Each detector maintains an internal 10 KB memory for logging debugging messages which are then printed out to the log file when an unexpected resource status report occurs. The buffer is a circular buffer such that if it fills before anything is printed out, it will be reused from the beginning and any existing data contained within the buffer will be overwritten and lost.

Each internal log message in the detector has an associated logging level. Only those messages which are lesser than or equal to the current log level setting will be added into the internal circular buffer. By default, only the internal messages marked with a debugging level of `1` are inserted into the buffer. The greater the value, the more debugging information is printed; however, the contents of logs may vary from detector to detector. The valid range of values is `1` to `9` (default value is `1`). This can be modified in the `hvw` command as follows:

1. Select the *Configuration-Edit-Global-Settings* menu.
2. Choose the *DetectorDetails* sub-menu.
3. Select *MemoryLogLevel*.

When an unexpected `Offline` or `Fault` resource state occurs, the debugging messages are printed from the circular buffer into the detector log file. The information is intended to help determine why the unexpected status report occurred. Because the circular buffer stores earlier logging messages, the log file will contain several DEBUG statements with dates prior to the last reported item appearing prior to printing out the circular buffer. The reason for keeping and printing the circular buffer is that a problem has occurred and with the aid of the debugging statements printed from the circular, it can be determined why the detector reported an unexpected resource state change.

7.9.2 Modifying levels while RMS is running

It is now possible to turn debug reporting on or off within the RMS Wizard detectors dynamically by using the `hvw` command as follows:

1. Select *Configuration-Edit-Global-Settings*.
2. Choose the *DetectorDetails* sub-menu.
3. Select the *DynamicDetectorLogging* menu item.

The default value is `0`, which means that debugging is turned off. By setting the value to something greater than zero, debugging is turned on. The greater the value, the more debugging information that is printed; however, the contents of logs may vary from detector to detector. The valid range of values is `1` to `9` (`0` means logging is turned off). Any modification to this setting takes affect the next time the configuration is activated.

The command actually creates the file *<RELIANT_LOG_PATH>*`/etc/`*wizardloglevel*, with its contents being the numerical value of the desired debug level. A value of zero in the file turns debugging off.

Alternatively, you can create the file *wizardloglevel* file manually. If the file exists, a default debugging level of `3` is used. The debug level can be modified by inserting a numerical value in the file.

i It is important to realize that by turning on debugging in this manner, all detectors will be affected and print out the additional debugging information.

Be aware that turning on the debugging levels in this manner should only be done when problems occur and for debugging purposes. Once any problems are resolved, debugging should again be turned off so as not to unnecessarily fill up the file system with extraneous information and cause the file system to fill.

7.10 PCS log files

The PCS log file is `/var/opt/SMAW/log/pcs.log.` The PCS log file may be useful to service personnel if an internal program error occurs. If the Trace Option is set several trace files are created in the `/var/opt/SMAW/SMAWpcs/trace` directory.Trace files may help service personnel diagnose internal problems with PCS.

7.10.1 Manual Script Execution

The expert user will occasionally want to execute an object's scripts (online, offline etc.) one at a time for diagnostic purposes. This functionality is referred to as Manual Script Execution (MSE).

MSE is called from different places in the PCS GUI and PCS CUI:

- In the PCS GUI this functionary is available from the *Draw Graph* window.
- In the PCS CUI this functionality is available in the *Advanced Menu* under the Manual Script Execution Tree.

The configuration must be generated for MSE to available. To invoke MSE the user must right-click on the desired RMS resource in the PCS GUI graph, or select the MSE menu item in the PCS CUI, and then select the desired script to execute. The output of the scripts is displayed in the log file `/var/opt/SMAW/log/pcs.log`.

7.11 RMS troubleshooting

When problems occur, RMS prints out meaningful error messages that will assist you in troubleshooting the cause. If no message is available, the following information may help you diagnose and correct some unusual problems:

- RMS dies immediately after being started.

At startup, the RMS base monitor exchanges its configuration checksum with the other base monitors on remote nodes. If the checksum of the starting base monitor matches the checksums from the remote nodes, the startup process continues. If the checksums do not match, then the RMS base monitor shuts down if all of the following conditions are true:

1. The base monitor has encountered a different checksum from a remote monitor within the initial startup period (defined by `HV_CHECKSUM_INTERVAL`).
2. There are no applications on this node that are `Online`, waiting, busy, or locked.
3. There are no online remote base monitors encountered by this base monitor.

Otherwise, the base monitor keeps running, but all remote monitors whose checksums do not match the local configuration checksum are considered to be `Offline`, so no message exchange is possible with these monitors, and no automatic or manual switchover will be possible between the local monitor and these remote monitors.

When different checksums are encountered, certain messages are placed in the switchlog explaining the situation.

Action:

Verify the problem by using `hvdisp -a` on the remote nodes to find out the actual configuration files. Compare the checksum of these configuration files. (The `hvdisp` command does not require root privilege.)

If the base monitor does not shut down on its own, but keeps running because one of the above conditions is not true, the system administrator may need to do the following:

1. Shut down certain base monitors.
2. Find out which configuration to run.
3. Distribute this file with `hvdist`.
4. Stop and restart RMS on the entire cluster so that all cluster nodes run the same configuration.

- RMS hangs after startup (processes are running, but `hvdisp` hangs)

This problem might occur if the local node is in the CF state LEFTCLUSTER from the point of view of the other (at least some other) cluster nodes.

Action:

Verify the problem by calling `cftool -n` on all cluster nodes to check for a possible `LEFTCLUSTER` state.

Call `cftool -k` to clear the `LEFTCLUSTER` state. RMS will continue to run as soon as the node has joined the cluster. No restart should be necessary.

- RMS loops (or even dies) shortly after being started.

  This problem could occur if the CIP configuration file `/etc/cip.cf` contains entries for the netmask. These entries are useless (not evaluated by CIP). From the RMS point of view these entries cannot be distinguished from IP addresses, which have the same format, so RMS will invoke a `gethostbyaddr`(). This normally does no harm, but in some unusual cases the OS may become confused.

  Action:

  Verify the problem by checking if netmask entries are present in `/etc/cip.cf`.

  Remove the netmask entries, and restart RMS.

- RMS detects a node failure (`network connection failed to host \...`), but does not even attempt to kill the node.

  This problem could occur if the failed node was already in a pending Wait state from an earlier failed kill request.

  i If a kill request fails, the `SysNode` remains in the Wait state until this state is manually cleared by the System Administrator.

  Action:

  Verify the problem by using `hvdisp -T SysNode` to see the states of all `SysNode` objects. (The `hvdisp` command does not require root privilege.)

  If you verify that a `SysNode` is in a pending Wait state, call `hvutil -o` <*SysNode*> or `hvutil -u` <*SysNode*>.

Caution

'`hvutil -u`' causes the surviving node to assume that the `SysNode` is actually dead, and it will invoke a failover immediately. If the node is still active, this may cause data corruption.

Caution

'`hvutil -o`' causes the surviving node to assume that the `SysNode` was alive the entire time. Therefore, it will continue assuming to be in sync with the remote `SysNode`. If this assumption is not true, this could cause unpredictable behaviors and, in a worst case scenario, data corruption.

The detector cycle time can be changed from its default value by using the `-w` option in `hvcm` command as '`hvcm -w` *<n>* `-c` *<config_file>*' where *n* is the new detector cycle time. This value must be greater than the value of `HV_CONNECT_TIMEOUT`.

- The RMS base monitor detects a loss of cluster heartbeat, but there is no indication as to the reason for the loss.

  RMS automatically invokes a tool that provides diagnostic information for this event.

Action:

The diagnostic tool performs the following actions:

– Invokes `truss(1)` on Solaris or `strace(1)` on Linux to trace the detector process

– Turns on full RMS and detector logging with the `-l0` (lowercase "L"-zero) option

– Gathers system and users times for the process

The `truss(1)`/`strace(1)` invocation and logging levels will be terminated after the number of seconds specified in the `ScriptTimeout` attribute. All information is stored in the `switchlog` file.

8 Non-fatal error messages

This chapter contains a detailed list of all non-fatal RMS error messages that appear in the switchlog. Most messages are accompanied by a description of the probable cause(s) and a suggested action to correct the problem. In some cases, the description or action is self-evident and no further information is necessary.

Some messages in the listings that follow contain words printed in *italics*. These words are placeholders for values, names, or strings that will be inserted in the actual message when the error occurs.

RMS error code description

A prefix in each message contains an error code and message number identifying the RMS component that detected the problem. You may need to provide this prefix to support engineers who are diagnosing your problem. The following list summarizes the possible error codes and the associated component:

`ADC:`	Admin configuration
`ADM:`	Admin, command, and detector queues
`BAS:`	Startup and configuration errors
`BM:`	Base monitor
`CML:`	Command line
`CRT:`	Contracts and contract jobs
`CTL:`	Controllers
`CUP:`	userApplication contracts
`DET:`	Detectors
`GEN:`	Generic detector
`INI:`	init script
`MIS:`	Miscellaneous
`NOD:`	Node detector
`QUE:`	Message queues
`SCR:`	Scripts
`SWT:`	Switch requests (hvswitch command)
`SYS:`	SysNode objects
`UAP:`	userApplication objects
`US:`	us files
`WLT:`	Wait list
`WRP:`	Wrappers

8.1 ADC: Admin configuration

- `(ADC, 1) Since this host <`*hostname*`> has been online for no more than` *time* `seconds and due to the previous error, it will shut down now.`

  time is the value of the environment variable `HV_CHECKSUM_INTERVAL`, if set, or 120 seconds otherwise. This message could appear when the checksums of the configurations of the local and the remote host are different, no more than *time* seconds have elapsed, and one of the following is true:
  - When the remote host is joining the cluster, and all the applications on the local host are either Offline or Faulted. RMS exits with exit code 60.
  - The configuration for the local host does not include the remote host, but the configuration for the remote host does include the local host. The local host *hostname* will shut down with exit code 60.

  Action:
  The local and the remote hosts are running different configurations. Make sure that both of them are running the same configuration.

- `(ADC, 2) Since not all of the applications are offline or faulted on this host <`*hostname*`>, and due to the previous error, it will remain online, but neither automatic nor manual switchover will be possible on this host until <`*detector*`> detector will report offline or faulted.`

  The checksums of the configurations of the local and the remote host are different, no more than the number of seconds determined by the value of the environment variable `HV_CHECKSUM_INTERVAL` have passed, and not all of the applications are offline or faulted. RMS will continue to remain online, but neither automatic nor manual switchover will be possible on this host until the detector *detector* reports offline or faulted.

  Action:
  Make sure that both the local and the remote host are running the same configuration.

- `(ADC, 3) Remote host <`*hostname*`> reported the checksum (`*remotechecksum*`) which is different from the local checksum (`*localchecksum*`).`

If the checksum of the configuration file reported by the remote host <*hostname*> is different from the checksum of the configuration file on the local host, this message will appear.

Action:

The most likely cause for this would be that the local host and the remote host are running configuration files that differ. Make sure that the local host and the remote host are running the same configuration file.

- `(ADC, 4) Host `<*hostname*>` is not in the local configuration.`

  This message is a result of the following problem: If the checksum reported by the remote host is different from that of the local host and if the configuration for the local host does not include the remote host's name, but the configuration for the remote host *hostname* includes the local host.

  Action:

  Make sure that the local and the remote host are running the same configuration.

- `(ADC, 5) Since this host `<*hostname*>` has been online for more than `*time*` seconds, and due to the previous error, it will remain online, but neither automatic nor manual switchover will be possible on this host until `<*detector*>` detector will report offline or faulted.`

  If the checksums of the configurations of the local and the remote host are different and if more than *time* seconds have elapsed since this host has gone online (*time* is the value of the environment variable `HV_CHECKSUM_INTERVAL`, if set or equal to 120 seconds if not), then RMS prints the above message.

  Action:

  Make sure that all the hosts in the cluster are running the same configuration file.

- `(ADC, 15) Global environment variable `<*envattribute*>` is not set in hvenv file.`

  This message is the result of RMS being unable to set the global environment variable <*envattribute*> because it has not been set in `hvenv`. *envattribute* can be any one of the following: `RELIANT_LOG_LIFE`, `RELIANT_SHUT_MIN_WAIT`, `HV_CHECKSUM_INTERVAL`, `HV_LOG_ACTION_THRESHOLD`, `HV_LOG_WARNING_THRESHOLD`, `HV_WAIT_CONFIG` or `HV_RCSTART`. This will eventually cause RMS to exit with exit code 1.

Action:
Set the value of the environment variable to an appropriate value.

- `(ADC, 17)` <*hostname*> `is not in the Wait state, hvutil -u request skipped!`

  When '`hvutil -u`' has been invoked on a node, if the `SysNode` for that node is not in the Wait State, then this message will appear (internal option).

  Action:
  If the '`hvutil -u`' was issued prematurely, then reissue the command once the node has reached the Wait state.

- `(ADC, 18) Local environment variable` <*envattribute*> `is not set in hvenv file.`

  If one of the local environment variables <*envattribute*> is not set in `hvenv`, this message is the result. *envattribute* can be any one of the following: `SCRIPTS_TIME_OUT`, `RELIANT_INITSCRIPT`, `RELIANT_STARTUP_PATH`, `HV_CONNECT_TIMEOUT`, `HV_MAXPROC` or `HV_SYSLOG_USE`. This will eventually cause RMS to exit with exit code 1.

  Action:
  Set the value of *envattribute* to an appropriate value.

- `(ADC, 20)` <*hostname*> `is not in the Wait state. hvutil -o request skipped!`

  The '`hvutil -o`' command has been invoked on a node, but its `SysNode` is not in the Wait State. (Internal option).

  Action:
  The '`hvutil -o`' was issued prematurely. Reissue the command after the `SysNode` has reached the Wait state.

- `(ADC, 25) Application` <*appname*> `is locked or busy, modification request skipped.`

`hvmod` has been invoked without the `-l` option, and the application is busy. Some other modification is already in progress, or some requests are being processed, or application contracting is ongoing.

Action:
Reissue the `hvmod` command when the application has completed the current switch request.

- `(ADC, 27) Dynamic modification failed.`

  Dynamic modification has failed. The exact reason for the failure is displayed in the message preceding this one.

  Action:
  Check the error messages occurring in the switchlog or prior to this message to find out the exact cause of the failure.

- `(ADC, 30) HV_WAIT_CONFIG value` <*seconds*> `is incorrect, using 120 instead.`

  If the value of the environment variable `HV_WAIT_CONFIG` is 0 or has not been set, the default value of 120 is used instead.

  Action:
  Set the value of `HV_WAIT_CONFIG` in `/opt/SMAW/SMAWRrms/bin/hvenv`.

- `(ADC, 31) Cannot get the NET_SEND_Q queue.`

  RMS uses the `NET_SEND_Q` queue for transmitting contract information. If there is some problem with this queue, the operation is aborted. The operation can be any one of the following: `hvrcp`, `hvcopy`.

  Action:
  Contact field support.

- `(ADC, 32) Message send failed during the file copy of file` <*filename*>`.`

  A error occurred while transferring file <*filename*> across the network.

  Action:
  Check if there are any problems with the network.

- `(ADC, 33) Dynamic modification timeout.`

The time taken for dynamic modification is greater than the timeout value. This timeout is equal to the value of the environment variable MODIFYTIMEOUTLIMIT if it is greater than 0 or else it is equal to 0 if the value of the environment variable is less than or equal to 0. If the environment variable itself is not defined then, the timeout value is 120 seconds by default.

Action:
Contact the field support.

- `(ADC, 34) Dynamic modification timeout during start up - bm will exit.`

  If the time taken for dynamic modification during bm startup is greater than the timeout value which is determined from the value of the environment variable MODIFYTIMEOUTLIMIT if it is greater than 0 or equal to 0 if the value of the environment variable is less than or equal to 0 or 120 seconds by default if the environment variable is not defined. RMS then exits with exit code 63.

  Action:
  Contact the field support.

- `(ADC, 35) Dynamic modification timeout, bm will exit.`

  Critical internal error.

  Action:
  Contact field support.

- `(ADC, 37) 75. Dynamic modification failed: cannot make a non-critical resource` <*resource*> `critical by changing its attribute MonitorOnly to 0 since this resource is not online while it belongs to an online application` <*appname*>`; switch the application offline before making this resource critical.`

  During dynamic modification, if there is an attempt to make a non-critical resource <*resource*> `MonitorOnly` while it is not online and the application <*appname*> is Online this message is the result along with dynamic modification aborting.

  Action:
  Switch the userApplication Offline before making the resource critical.

- `(ADC, 38) 76. Dynamic modification failed: application` *<appname>* `has no children, or its children are not valid resources.`

  If RMS finds that the userApplication *<appname>* will have no children while performing dynamic modification, this message is printed out to the switchlog and dynamic modification is aborted.

  Action:
  Make sure that the userApplication has valid children while performing dynamic modification.

- `(ADC, 39) The putenv() has failed (`*failurereason*`)`

  The wizards use the environment variable `HVMOD_HOST` during dynamic modification. This variable holds the name of the host on which `hvmod` has been invoked. If this variable cannot be set with the function `putenv()`, then this message is printed to the switchlog along with the reason *failurereason*.

  Action:
  Check the reason *failurereason* in the switchlog to find out why this operation has failed and take corrective action based on this.

- `(ADC, 41) The Wizard action failed (`*command*`)`

  Wizards make use of an action file during `hvmod`. If the execution of this action file (*command*) has failed due to the process exiting by using an exit call, this message is printed out to the switchlog along with the reason for this failure printed out.

  Action:
  Check the switchlog for finding the reason for this failure and rectify it before reissuing the `hvmod` command.

- `(ADC, 43) The file transfer for` *<filename>* `failed in "`*command*`". The dynamic modification will be aborted.`

  During dynamic modification, files containing modification information are transferred between the hosts of the cluster. If, for any reason, a file transfer fails, the dynamic modification is aborted.

  Action:
  Make sure that host and cluster conditions are such that *command* can be safely executed.

- `(ADC, 44) The file transfer for <`*filename*`> failed in "`*command*`". The join will be aborted.`

  When a host joins a cluster, it receives a cluster configuration file. If, for any reason, a file transfer fails, the dynamic modification is aborted.

  Action:
  Make sure that host and cluster conditions are such that *command* can be safely executed.

- `(ADC, 45) The file transfer for <`*filename*`> failed in "`*command*`" with errno <`*errno*`> - `*errorreason*`. The dynamic modification will be aborted.`

  During dynamic modification, files containing modification information are transferred between the hosts of the cluster. If, for any reason, a file transfer fails, the dynamic modification is aborted. A specific reason for this failure is referred to by the OS error code ERRNO and its explanation in ERRORREASON.

  Action:
  Make sure that host and cluster conditions are such that *command* can be safely executed.

- `(ADC, 46) The file transfer for <`*filename*`> failed with unequal write byte count, expected `*expectedvalue*` actual `*actualvalue*`. The dynamic modification will be aborted.`

  During dynamic modification, files containing modification information are transferred between the hosts of the cluster. During the transfer, RMS keeps track of the integrity of the transferred data by counting the bytes transferred. This count can be incorrect if the transfer process is broken or interrupted.

  Action:
  Make sure that host, cluster and network conditions are such that *command* can be safely executed.

- `(ADC, 47) RCP fail: can't open file `*filename*`.`

  If the file <*filename*> that has been specified as the file to be copied from the local host to the remote host cannot be opened for reading, this message is the result.

  Action:

Make sure that the file <*filename*> is readable.

- `(ADC, 48) RCP fail: fseek errno` ***errno***`.`

  During a file transfer between the hosts, RMS encountered a problem indicated by the OS error code ERRNO.

  Action:
  Make sure that the host, cluster and network conditions are such that file transfer proceeds without errors.

- `(ADC, 49) Error checking hvdisp temporary file` <*filename*>`, errno` <*errno*>`, hvdisp process pid` <*processid*> `is restarted.`

  The RMS base monitor periodically checks the integrity and size of the temporary file used to transfer configuration data to the `hvdisp` process. If this file cannot be checked, then `hvdisp` process is restarted automatically, though some data may be lost and not displayed at this time. Specific OS error code for the error encountered is displayed in ERRNO.

  Action:
  Make sure that the host conditions are such that the temporary file can be checked. Sometimes, you may need to restart the `hvdisp` process by hand.

- `(ADC, 57) An error occurred while writing out the RMS configuration for the joining host. The hvjoin operation is aborted.`

  When a remote host joins a cluster, this host attempts to dump its own configuration for a subsequent transfer to the remote host. If the configuration cannot be saved, the `hvjoin` operation is aborted.

  Action:
  One of the previous messages contain a detailed explanation about the error occurring while saving the configuration. Correct the host environment according to the explanation, or contact field support.

- `(ADC, 58) Failed to prepare configuration files for transfer to a joining host. Command used` <*command*>`.`

When a remote host joins a cluster, this host attempts to prepare its own configuration for a subsequent transfer to the remote host. For that, it uses the command <*command*>. If the <*command*> fails, the `hvjoin` operation is aborted.

Action:
Contact field support.

- `(ADC, 59) Failed to store remote configuration files on this host. Command used` <*command*>`.`

  When this host joins a cluster, this host attempts to store remote configuration files for a subsequent dynamic modification on this host. For that, it uses the command <*command*>. If the <*command*> fails, the `hvjoin` operation is aborted.

  Action:
  Contact field support.

- `(ADC, 60) Failed to compress file` <*file*>`. Command used` <*command*>`.`

  File transfer is a part of some RMS operations such as dynamic modification and `hvjoin`. Before transferring a file <*file*> to a remote host, it must be compressed with the command <*command*>. If the <*command*> fails, the operation that requires the file transfer is aborted.

  Action:
  Contact field support.

- `(ADC, 61) Failed to shut down RMS on host` <*host*>`.`

  While performing RMS cluster-wide shutdown, RMS on host <*host*> failed to shut down.

  Action:
  Contact field support.

- `(ADC, 62) Failed to shut down RMS on this host, attempting to exit RMS.`

  While performing RMS clusterwide shutdown, RMS on this host failed to shut down. Another attempt to shut down this host is automatically initiated.

Action:
Contact field support.

- `(ADC, 63) Error` *<errno>* `while reading file` *<file>*`, reason:` *<reason>*`.`

  While reading file <*file*>, an error <*errno*> occurred explained by <*reason*>. File reading errors may occur during dynamic modification, or during `hvjoin` operation.

  Action:
  Contact field support.

- `(ADC, 68) Error` *<errno>* `while opening file` *<file>*`, reason:` *<reason>*`.`

  While opening file <*file*>, an error <*errno*> occurred explained by <*reason*>. File open errors may occur during dynamic modification.

  Action:
  Verify the file existence and reissue dynamic modification request.

- `(ADC, 70) Message sequence # is out of sync - File transfer of file` *<filename>* `has failed.`

  Critical internal error.

  Action:
  Contact field support.

8.2 ADM: Admin, command, and detector queues

- `(ADM, 3) 31. Dynamic modification failed: some resource(s) supposed to come offline failed.`

  During dynamic modification when new resource(s) that are to be added to a parent object that is offline cannot be brought offline, this message is the result.

  Action:
  Make sure the new resource(s) can be brought to the offline state and reissue the `hvmod` command.

- `(ADM, 4) 30. Dynamic modification failed: some resource(s) supposed to come online failed.`

  During dynamic modification when new resource(s) that are to be added to a parent object that is online by executing the online scripts cannot be brought online, dynamic modification is aborted.

  Action:
  Make sure the new resource(s) can be brought to the online state and reissue the `hvmod` command.

- `(ADM, 5) 17. Dynamic modification failed: object` <*object*> `is not linked to any application.`

  During dynamic modification, if there is an attempt to add an object <*object*> that does not have a parent (and hence not linked to any `userApplication`), this message is printed and dynamic modification is aborted.

  Action:
  Make sure that every object being added during dynamic modification is linked to a `userApplication`.

- `(ADM, 6) 36. Dynamic modification failed: cannot add new resource` <*resource*> `since another existing resource with this name will remain in the configuration.`

  When RMS receives a directive to add a new resource <*resource*> with the same name as that of an existing resource, this message is printed out to the switchlog and dynamic modification aborts.

  Action:
  Make sure that when adding a new resource, its name does not match the name of any other existing resource.

- `(ADM, 7) 35. Dynamic modification failed: cannot add new resource` <*resource*> `since another existing resource with this name will not be deleted.`

  When RMS receives a directive to add a new resource <*resource*> with the name of an existing resource, it prints out this message and dynamic modification aborts.

  Action:

Make sure that when adding a new resource, its name does not match the name of any other existing resource.

- `(ADM, 8) 29. Dynamic modification failed: cycle of length` *<cyclelength>* `detected in resource` *<resource>* `--` *<cycle>*`.`

  In the overall structure of the graph of the RMS resources, no cycles are allowed along the chains of parent/child links. If this is not the case then dynamic modification fails and the message specified above will be printed to the switchlog.

  Action:
  Get rid of the cycles.

- `(ADM, 9) 34. Dynamic modification failed: cannot modify resource` *<resource>* `since it is going to be deleted.`

  Since, deleting a resource causes all its children with no other parents to get deleted as well, deleting a resource and then modifying the attributes of the deleted resource or a child of that resource that has no other parents leads to dynamic modification being aborted and the message being printed to the switchlog.

  Action:
  While performing dynamic modification of a resource make sure that the resource that is being modified has not been deleted.

- `(ADM, 11) 37. Dynamic modification failed: cannot delete object` *<resource>* `since it is a descendant of another object that is going to be deleted.`

  When there is an attempt to delete a child object when the parent object has been deleted, the above message will appear in the switchlog and dynamic modification aborted.

  Action:
  Make sure that when an object is being deleted explicitly, its parents have not already been deleted because that means this object has also been deleted.

- `(ADM, 12) 38. Dynamic modification failed: cannot delete` *<resource>* `since its children will be deleted.`

When there is an attempt to delete a resource <*resource*> whose children have already been deleted, the above message will appear in the switchlog and dynamic modification aborted.

Action:
Make sure that when a resource is being deleted explicitly, its children have not already been deleted.

- `(ADM, 13) 52. dynamic modification failed: object` <*resource*> `is in state` <*state*> `while needs to be in one of stateOnline, stateStandby, stateOffline, stateFaulted, or stateUnknown.`

  Every resource has to be in either one of the states: stateOnline, stateOffline, stateFaulted, stateUnknown or stateStandby. If the resource <*resource*> is not in any of the states mentioned above, it prints the above message and dynamic modification is aborted. Theoretically this is not possible.

  Action:
  Contact field support.

- `(ADM, 14) 48. Dynamic modification failed: cannot link to or unlink from an application` <*appname*>`.`

  If the parent of the resource is a `userApplication`, then linking to or unlinking a child from that parent is not possible. If there is an attempt to perform this, then the above message will be printed to the switchlog and dynamic modification will be aborted.

  Action:
  Do not link or unlink a resource from a `userApplication`.

- `(ADM, 15) 41. Dynamic modification failed: parent object` <*parentobject*> `is not a resource.`

  When RMS gets a directive to link existing resources during dynamic modification, if the parent object <*parentobject*> to which the child object is being linked is not a resource, then dynamic modification fails and this message is printed.

  Action:
  Make sure that while linking 2 objects, the parent of the child object is a resource.

- `(ADM, 16) 42. Dynamic modification failed: child object` *<childobject>* `is not a resource.`

  When RMS gets a directive to link existing resources during dynamic modification, if the child object *<childobject>* that is being linked to a parent object is not a resource, then dynamic modification fails and this message is printed.

  Action:

  Make sure that while linking 2 objects, the child of the parent object is a resource.

- `(ADM, 17) 43. Dynamic modification failed: cannot link parent` *<parentobject>* `and child` *<childobject>* `since they are already linked.`

  Trying to link a parent *<parentobject>* and a child *<childobject>* which are already linked results in this message. Dynamic modification will be aborted.

  Action:

  While trying to perform dynamic modification make sure that the parent and the child that are to be linked are not already linked.

- `(ADM, 18) 49. Dynamic modification failed: cannot link a faulted child` *<childobject>* `to parent` *<parentobject>* `which is not faulted.`

  While creating a new link between 2 existing objects, during dynamic modification, a faulted child *<childobject>* cannot be linked to a parent *<parentobject>* that is not faulted. The child first needs to be brought to the state of the parent. If this condition is violated, the aforementioned message will be printed to the switchlog. Dynamic modification is aborted.

  Action:

  Bring the faulted child to the state of the parent before linking them.

- `(ADM, 19) 50. Dynamic modification failed: cannot link child` *<childobject>* `which is not online to online parent` *<parentobject>*`.`

  While linking 2 existing objects during dynamic modification, the combination of states parent Online and child not Online is not allowed. When this happens, dynamic modification is aborted and a message is printed to the switchlog.

Action:

The child <*childobject*> first needs to be brought to the online state before linking it to the online parent <*parentobject*>.

- `(ADM, 20) 51. Dynamic modification failed: cannot link child` <*childobject*> `which is neither offline nor standby to offline or standby parent` <*parentobject*>`.`

  Any attempt to link 2 existing objects in which the child is neither in the Offline nor the Standby state and the parent is in the Offline or Standby states is prohibited and results in the message being written to the switchlog. Dynamic modification is aborted.

  Action:

  The child needs to be first brought to offline or standby state before linking it to the parent that is in offline or standby state.

- `(ADM, 21) 44. Dynamic modification failed: Cannot unlink parent` <*parentobject*> `and child` <*childobject*> `since they are not linked.`

  Trying to unlink object <*parentobject*> from object <*childobject*> when they are not already linked results in this message with dynamic modification aborted.

  Action:

  If you want to unlink 2 objects make sure that they share a parent child relationship.

- `(ADM, 22) 46. Dynamic modification failed: child` <*childobject*> `will be unlinked but not linked back to any of the applications.`

  Unlinking a child <*childobject*> so that no links remain linking it to any `userApplication` is not allowed.

  Action:

  Make sure that the child is still linked to a `userApplication`.

- `(ADM, 23) 47. Dynamic modification failed: sanity check did not pass for linked or unlinked objects.`

  Dynamic modification performs some sanity checks to ensure that all of the following are true:

- The `HostName` attribute is present only for children of `userApplication` objects.
- The child of a `userApplication` does not have another parent.
- Each object belongs to only one `userApplication`.
- Leaf objects have detectors.
- Leaf objects that have the `DeviceName` attribute have it set to a valid value.
- The length of the attribute `rName` for the leaf objects is smaller than the maximum.
- There are no duplicate lines in the `hvgdstartup` file.
- The `kind` argument for the detector in the `hvgdstartup` is specified.
- All detectors can be loaded.
- A valid value has been specified for the `rKind` attribute.
- The `ScriptTimeout` value is greater than the detector cycle time.
- No objects are `and` and `or` at the same time.
- `ClusterExclusive` and `LieOffline`, which are mutually exclusive, are not used together.

If some of these sanity checks fail, then this message will be printed and dynamic modification is aborted. A FATAL message is also printed to the switchlog with more details as to why the sanity check failed.

Action:

Make sure that the sanity checks mentioned above pass.

- `(ADM, 24) 45. Dynamic modification failed: object` *<object>* `that is going to be linked or unlinked will be either deleted, or unlinked from all applications.`

Any attempt to perform the operations of deleting an object <*object*> from the RMS resource graph and then trying to unlink it from its parent object or vice versa results in dynamic modification being aborted and the above message being printed out to the switchlog.

Action:

Make sure that the operations of deletion and unlinking are not performed on an object at the same time.

- `(ADM, 25) 1. Dynamic modification failed: parent object` *<parentobject>* `is absent.`

When a new object is being added to an existing configuration, it should have an existing object <*parentobject*> as its parent, if not then, dynamic modification is aborted and the message is printed to the switchlog.

Action:
Make sure that the parent specified for a new object that is being added is existent.

- `(ADM, 26) 18. Dynamic modification failed: parent object` <*parentobject*> `is neither a resource nor an application.`

  When a new object is being added to an existing configuration, if the parent object <*parentobject*> that has been specified is not a resource, it leads to dynamic modification aborting and the message being printed. Dynamic modification is aborted.

  Action:
  Make sure that the parent object specified for a new object is a resource.

- `(ADM, 27) 2. Dynamic modification failed -- child object` <*childobject*> `is absent.`

  Any attempt to link to a child object <*childobject*> that is non-existent leads to this message and dynamic modification aborts.

  Action:
  Make sure that the child object to be linked to exists.

- `(ADM, 28) 19. Dynamic modification failed: child object` <*childobject*> `is not a resource.`

  When a new object <*childobject*> being added to an existing configuration is not a resource, this message is the result and dynamic modification aborts.

  Action:
  Make sure that the child object specified is a resource.

- `(ADM, 29) 3. Dynamic modification failed -- parent object` <*parentobject*> `is absent.`

  Action:
  Critical error. Contact field support.

- `(ADM, 30) 20. Dynamic modification failed: parent object` <*parentobject*> `is not a resource.`

During dynamic modification if there is a request to add a new parent object <*parentobject*> that is not a resource, this message occurs and dynamic modification aborts.

Action:
Make sure that the object being added as a parent object is a resource.

- `(ADM, 31) 4. Dynamic modification failed: child object` <*childobject*> `is absent.`

  As part of dynamic modification, if the specified child object <*childobject*> does not exist, then this message is the result and dynamic modification is aborted.

  Action:
  Make sure that the child object that has been specified exists.

- `(ADM, 32) 21. Dynamic modification failed: child object` <*childobject*> `is not a resource.`

  When adding a new object to the RMS resource graph, if the child <*childobject*> of this new object is not a resource, dynamic modification aborts.

  Action:
  Make sure that when adding a new object, its child is a resource.

- `(ADM, 33) 5. Dynamic modification failed: object` <*object*> `cannot be deleted since either it is absent or it is not a resource.`

  If RMS gets a directive to delete an object <*object*> that is either non-existent or not a resource, this message is the result along with the failure of dynamic modification.

  Action:
  Make sure that you don't try to delete an object that does not exist.

- `(ADM, 34) 22. Dynamic modification failed: deleted object` <*object*> `is neither a resource nor an application nor a host.`

  An object deleted during dynamic modification is neither a resource type object, nor a `userApplication` nor a `SysNode` object. Only resources, applications and hosts (`SysNode` objects) can be deleted during dynamic modification.

Action:
Do not delete this object, or delete another object.

- `(ADM, 37) 6. Dynamic modification failed: resource` <*object*> `cannot be brought online and offline/standby at the same time.`

  When a resource <*object*> is added to an existing RMS resource graph and it is linked as a child to two parent objects, one of which is online and the other offline/standby, this message is the result: a child object needs to be brought to the state of its parent.

  Action:
  Make sure that both the parents of the resource to be added are in the same state before adding it.

- `(ADM, 38) 7. Dynamic modification failed: existing parent resource` <*parentobject*> `is in state` <*state*> `but needs to be in one of stateOnline, stateStandby, stateOffline, state-Faulted, or stateUnknown.`

  During dynamic modification, if the state <*state*> of a parent resource <*parentobject*> is not one of the states stateOnline, stateOffline, state-Faulted, or stateUnknown, dynamic modification aborts.

  Action:
  Make sure that the state of the parent resource is one of the states mentioned above.

- `(ADM, 39) 28. Dynamic modification failed: new resource` *object* `which is a child of application` <*userApplication*> `has its HostName` <*hostname*> `the same as another child of application` <*appname*>`.`

  When a new object *object* is being added as a child of <*appname*> and the value of its `HostName` attribute is the same as the value of the `HostName` attribute of an existing child of <*appname*>, this message is the result and dynamic modification aborts after this.

  Action:
  Make sure that the `HostName` attribute of an object that is being added to `userApplication` is different from the values of the `HostName` attributes of other first level children of *appname*.

- `(ADM, 40) 25. Dynamic modification failed: a new child` <*childobject*> `of existing application` <*appname*> `does not have its HostName set to a name of any sysnode.`

  When a new child object <*childobject*> is added to an application <*appname*> during dynamic modification, if the `HostName` attribute is missing for this object, this message is the result, with dynamic modification aborting.

  Action:
  The first level object under *appname* must have a `HostName` attribute.

- `(ADM, 41) 8. Dynamic modification failed: existing child` <*childobject*> `is not online, but needs to be linked with` <*parentobject*> `which is supposed to be brought online.`

  If both the parent <*parentobject*> and the child <*childobject*> have detectors associated with them, if the state of the child is not online, but it needs to be linked to the parent which is supposed to be online, then this message will be printed and dynamic modification aborted.

  Action:
  Make sure that the parent and the child are in a similar state.

- `(ADM, 42) 9. Dynamic modification failed: existing child` <*childobject*> `is online, but needs to be linked with` <*parentobject*> `which is supposed to be brought offline.`

  Trying to link a child <*childobject*> that is online to a parent object, which is supposed to go offline is not allowed, and dynamic modification aborts.

  Action:
  Make sure that the parent and the child are in a similar state.

- `(ADM, 43) 10. Dynamic modification failed: linking the same resource` <*childobject*> `to different applications` <*userapplication1*> `and` <*userapplication2*>`.`

  When RMS gets a directive to add a new child object <*childobject*> having as parent and child resources belonging to different applications <*userapplication1*> and <*userapplication2*>, the above message is printed and dynamic modification aborts.

  Action:

When adding a new resource make sure that it does not have as its parent and children, resources belonging to different applications.

- `(ADM, 44) 11. Dynamic modification failed: object` *<object>* `does not have an existing parent.`

  Any attempt to create an object <*object*> that does not have an existing parent leads to this message and dynamic modification aborts.

  Action:

  Make sure that the object <*object*> has an existing object as its parent.

- `(ADM, 45) 55. Dynamic modification failed: HostName is absent or invalid for resource` *<object>*`.`

  If the HostName attribute of object <*object*> is an invalid value then this message occurs and dynamic modification is aborted. If the HostName attribute is missing, (ADM, 40) will take care of it.

  Action:

  Set the HostName attribute of resource <*object*> to the name of a valid `SysNode`.

- `(ADM, 46) 12. Dynamic modification failed: linking the same resource` *<object>* `to different applications` *<appname1>* `and` *<appname2>*`.`

  RMS received a directive to add a new child object <*object*> by linking it to parent objects belonging to different applications <*appname1*> and <*appname2*>. Dynamic modification is aborted.

  Action:

  When adding a new child resource, make sure that it does not have as its parents resources belonging to different applications.

- `(ADM, 47) 23. Dynamic modification failed: parent object` *<parentobject>* `belongs to a deleted application.`

  Any attempt to add a new node having as its parent <*parentobject*> fails if the parent <*parentobject*> is the child of an object that has been deleted, because deleting an object automatically causes its children to be deleted as well if they don't have any other parents. This causes dynamic modification to fail.

  Action:

When adding a new object make sure that its parent has not already been deleted.

- `(ADM, 48) 24. Dynamic modification failed: child object` <*childobject*> `belongs to a deleted application.`

  Any attempt to delete an object <*childobject*> belonging to a deleted application elicits this response from RMS because deleting an application automatically causes all its children to be deleted as well.

  Action:
  Do not try to delete an object belonging to an already deleted application.

- `(ADM, 49) 24. Dynamic modification failed: deleted object` <*objectname*> `belongs to a deleted application.`

  Any attempt to delete an object <*objectname*> that belongs to a deleted application leads to this error because deleting an application deletes all its children including <*objectname*>.

  Action:
  Make sure that before an object is deleted, it does not belong to an application that is being deleted.

- `(ADM, 50) 40. Dynamic modification failed: cannot delete object` <*object*> `since it is a descendant of a new object.`

  When RMS gets a directive to delete an object <*object*>, which is a descendant of a new object, dynamic modification aborts and this message is the result.

  Action:
  Make sure that when an object is being deleted, it is not a descendant of a new object.

- `(ADM, 51) 15. Dynamic modification failed: cannot link to child` <*childobject*> `since it will be deleted.`

  When RMS gets a directive to link to a child <*childobject*> that is going to be deleted, dynamic modification aborts.

  Action:
  Do not link to a child object, which is to be deleted.

- `(ADM, 52) 16. Dynamic modification failed: cannot link to parent` *<parentobject>* `since it will be deleted as a result of deletion of object` *<object>*`.`

  If there is an attempt to delete an object *<object>* and use its descendants (which should be deleted as a result of deleting the parent) as the parent for a new resource that is being added to the RMS resource graph, this error message is printed and dynamic modification aborts.

  Action:
  Do not attempt to delete an object and use its descendant as the parent for a new resource.

- `(ADM, 53) 26. Dynamic modification failed:` *<node>* `is absent.`

  Trying to modify the attribute of a node *<node>* which is absent leads to this error and dynamic modification aborts.

  Action:
  Modify the attributes of an existing node.

- `(ADM, 54) 27. Dynamic modification failed: NODE` *<object>*`, attribute` *<attribute>* `is invalid.`

  When RMS receives a directive to modify a node *<object>* with attribute *<attribute>* that has an invalid value, this message is the result and dynamic modification aborts.

  Action:
  Specify a valid value for the attribute *<attribute>*.

- `(ADM, 55) Cannot create admin queue.`

  RMS uses Unix queues internally for interprocess communication. Admin queue is one such queue that is used for communication between RMS and other utilities like `hvutil`, `hvmod`, `hvshut`, `hvswitch` and `hvdisp`. If RMS cannot create this queue due to some reason, RMS exits with exit code 50.

  Action:
  Restart RMS

- `(ADM, 57) hvdisp - open failed -` *filename*`.`

If RMS is unable to open the file `/opt/SMAW/SMAWRrms/locks/.rms.`*<pid>* for writing when `hvdisp` has been invoked, this message is printed out.

Action:

Verify that the directory `/opt/SMAW/SMAWRrms/locks` exists and allows files to be created (correct permissions, free space in the file system, free inodes). If one of these problems exists, fix it via the appropriate administrator operation. If none of these problems apply, but the RMS failure still occurs, contact RMS support.

- `(ADM, 58) hvdisp - open failed -` *filename* `:` *errormsg*`.`

  When `hvdisp` is unable to open the file *file* (`/opt/SMAW/SMAWRrms/locks/.rms.`*<pid>*) for writing, it prints out the reason *errormsg*.

  Action:

  Verify that the directory `/opt/SMAW/SMAWRrms/locks` exists and allows files to be created (correct permissions, free space in the file system, free inodes). If one of these problems exists, fix it via the appropriate administrator operation. If none of these problems apply, but the RMS failure still occurs, contact RMS support.

- `(ADM, 59)` *appname*`: modification is in progress, switch request skipped.`

  This message is printed to the switchlog because commands like `hvswitch`, `hvutil` and `hvshut` cannot run in parallel with a non local `hvmod`.

  Action:

  Make sure that before a `hvswitch` is performed, `hvmod` is not operating on *appname*.

- `(ADM, 60)` *<resource>* `is not a userApplication object, switch request skipped!`

  While performing a switch, `hvswitch` requires a `userApplication` as its argument. If the resource *<resource>* is not a `userApplication`, this message is the result.

  Action:

  Check the man page for `hvswitch` for usage information.

- `(ADM, 62) The attribute <ShutdownScript> may not be specified for object <`*object*`>.`

  The attribute ShutdownScript is a hidden attribute within a `SysNode`. The RMS base monitor automatically defines its value -- users cannot change it in any way.

  Action:
  Do not attempt to change the built-in value of the ShutdownScript attribute.

- `(ADM, 63) System name <`*sysnode*`> is unknown.`

  This message can occur in these scenarios:
  – The name of the `SysNode` specified in `hvswitch` is not included in the current configuration. ('`hvswitch [-f]` *appname* [*sysnode*]')
  – The name of the `SysNode` specified for '`hvshut -s` *sysnode*' is not a valid one, i.e., *sysnode* is not included in the current configuration.
  – The name of the `SysNode` specified for '`hvutil -ou`' is unknown (hidden options).

  Action:
  Specify a `SysNode` that is included in the current configuration, i.e., appears in the *configname*`.us` file.

- `(ADM, 67)` *sysnode* `Cannot shut down.`

  This message could appear if '`hvshut -a`' was invoked and not all of the nodes replied with an acknowledgement.

  Action:
  Login to the remote hosts. If RMS is still running, perform '`hvutil -f` <*appname*>' to shut down each application one at a time. If this fails, refer to the `switchlog` and <*appname*>`log` files to find the reason for the problem. If all applications have been shut down correctly, perform a forced RMS shutdown with '`hvshut -f`'. Report the problem to RMS support.

- `(ADM, 70) NOT ready to shut down.`

  The reason for this message is:
  If the node on which '`hvshut -a`' has been invoked is not yet ready to be shut down because the application is busy on the node.

  Action:

Wait until the ongoing action (e.g. switchover, dynamic reconfiguration) has terminated.

- `(ADM, 75) 57. Dynamic modification failed: child` <*resource*> `of userApplication object` <*appname*> `has HostName attribute` <*hostname*> `common with other children of the same userApplication.`

  This message occurs if the RMS internal sanity-check functions detect a severe configuration problem. This message should not occur if the configuration has been set up using RMS configuration wizards.

  Action:
  Contact field support.

- `(ADM, 76) Modification of attribute` <*attribute*> `is not allowed within existing object` <*object*>`.`

  The attribute <*attribute*> is constant and can only be set in a configuration file.

  Action:
  Make sure that there is no attempt to modify <*attribute*> within <*object*>.

- `(ADM, 77) 58. Dynamic modification failed: cannot delete object` *object* `since its state is currently being asserted.`

  This message can appear in the switchlog if dynamic modification is being performed on an object that is being asserted.

  Action:
  Perform the modification after the assertion has been fulfilled.

- `(ADM, 78) 59. Dynamic modification failed: PriorityList` <*prioritylist*> `does not include all the hosts where the application` <*appname*> `may become Online. Make sure that PriorityList contains all hosts from the HostName attribute of the application's children.`

  Set `PriorityList` for <*appname*> to include all the host names from the `HostName` attribute of the application's children.

  Action:
  No duplicate host names should be present in the `PriorityList`.

- `(ADM, 79) 60. Dynamic modification failed: PriorityList` <*prioritylist*> `includes hosts where the application` <*appname*> `cannot go Online. Make sure PriorityList contains only hosts from the HostName attributes of the application's children.`

  The `HostName` attribute of one or more of the children specifies hosts that are not in the parent's `PriorityList` attribute.

  Action:

  Set the `PriorityList` attribute of <*appname*> to include all the host names listed in the `HostName` attributes of the application's children. No duplicate host names should be present in the `PriorityList`.

- `(ADM, 81) 61. Dynamic modification failed: application` <*appname*> `may not have more than` <*maxcontroller*> `parent controllers as specified in its attribute MaxControllers.`

  If <*appname*> uses more parent controllers than specified by the attribute `MaxControllers` (<*maxcontroller*>), this message is the result and dynamic modification aborts.

  Action:

  Make sure that the number of parent controllers used by an application is less than the number specified as part of the `MaxControllers` attribute, or modify `MaxControllers` to increase the number.

- `(ADM, 82) 62. Dynamic modification failed: cannot delete SysNode` <*sysnode*> `unless its state is one of Unknown, Wait, Offline or Faulted.`

  This message may appear in the switchlog if there is an attempt to delete a `SysNode` from a running configuration if this `SysNode` is not in one of the states: Unknown, Offline, Wait or Faulted.

  Action:

  Shut down RMS on that host and do the deletion.

- `(ADM, 83) 63. Dynamic modification failed: cannot delete SysNode` <*sysnode*> `since this RMS monitor is running on this SysNode.`

  During dynamic modification the local `SysNode` <*sysnode*> was going to be deleted.

  Action:

Make sure dynamic modification does not contain '`delete` *sysnode*;' where *sysnode* is the name of the local node.

- `(ADM, 84) 64. Dynamic modification failed: cannot add SysNode <`*sysnode*`> since its name is not valid.`

  This message appears in the switchlog if the name <*sysnode*> specified as part of the dynamic modification is not resolvable to any known host name.

  Action:
  Specify a host name that is resolvable to a network address.

- `(ADM, 85) 65. Dynamic modification failed: timeout expired, timeout symbol is <`*symbol*`>.`

  If the dynamic modification takes too much time, this message is the result.

  Action:
  Make sure that the network connection between the hosts is functional, and also verify that the scripts from newly added resources do not take too much time to execute, or that dynamic modification does not add too many new nodes, or that the modification file is too big or too complex.

- `(ADM, 86) 66. Dynamic modification failed: application <`*appname*`> cannot be deleted since it is controlled by the controller <`*controller*`>.`

  A controlled application <*appname*> cannot be deleted while its controller <*controller*> retains the application's name in its `Resource` attribute.

  Action:
  Remove the name of the deleted application from the controller's `Resource` attribute, or add a new application with the same name, or delete the controller together with its controlled application, or change the controller's `NullDetector` attribute to 1.

- `(ADM, 87) 67. Dynamic modification failed: only local attributes such as ScriptTimeout, DetectorStartScript, NullDetector or MonitorOnly can be modified during local modification ("'hvmod -l'").`

The reason for this message is that only the modification of local attributes is allowed during local modification.

Action:
Make a non-local modification, or modify different attributes.

- `(ADM, 88) 68. Dynamic modification failed: attribute` <*attribute*> `is modified more than once for object` <*object*>`.`

  This message may appear because an attribute of a particular object can be modified only once in the same modification file, but <*attribute*> has been modified more than once for <%*object*>.

  Action:
  Modify the attribute only once per object.

- `(ADM, 89) 69. Dynamic modification failed: cannot rename existing object` <*sysnode*> `to` <*othersysnode*> `because either there is no object named` <*sysnode*>`, or another object with the name` <*othersysnode*> `already exists, or a new object with that name is being added, or the object is not a resource, or it is a SysNode, or it is a controlled application which state will not be compatible with its controller.`

  This message appears when we try to rename an existing object <*sysnode*> to other node <*othersysnode*> but one of the following conditions was encountered:
  <*othersysnode*> is not a valid name.
  <*othersysnode*> is already used by some other host in the cluster.
  <*othersysnode*> is not a resource.
  <*othersysnode*> is a controlled application.

  Action:
  Choose another valid host name.

- `(ADM, 90) 70. Dynamic modification failed: cannot change attribute Resource of the controller object` <*controllernode*> `from` <*oldresource*> `to` <*newresource*> `because some of` <*oldresource*> `are going to be deleted.`

  This message appears when the user tries to rename a resource that is controlled by a controller object and is going to be deleted.

  Action:
  Make sure deleted applications are not referred to from any controller.

- `(ADM, 91) 71. Dynamic modification failed: controller` *<controller>* `has its Resource attribute set to` *<resource>*`, but application named` *<appname>* `is going to be deleted.`

  This message appears when the user tries to control a resource <*resource*> with a controller <*controller*> but the application associated with that resource is going to be deleted.

  Action:

  Make sure the controller's `Resource` attribute does not refer to a deleted application.

- `(ADM, 95) Cannot retrieve information about command line used when starting RMS. Start on remote host must be skipped. Please start RMS manually on remote hosts.`

  This message is the result of starting RMS with the `-a` option but due to some internal error RMS could not be started on the remote host. Critical internal error.

  Action:

  Contact field support. For temporary workaround, try again or start RMS manually on each host.

- `(ADM, 96) Remote startup of RMS failed` *<startupcommand>*`. Reason:` *errorreason*`.`

  When RMS cannot be started on remote hosts because the command <*startupcommand*> failed.

  Action:

  This may occur when some of the hosts are not reachable or the network is down.

- `(ADM, 98) 72. Dynamic modification failed: controller` *<controller>* `has its Resource attribute set to` *<resource>*`, but some of the controlled applications from this list do not exist.`

  This message appears when the controller node was not able to find the applications controlled by it with the applications running on the host.

  Action:

  Correct your modification file so that the controllers refer only to the existing application.

- `(ADM, 99) 73. Dynamic modification failed: cannot change attribute Resource of the controller object` *<controller>* `from` *<oldresource>* `to` *<newresource>* `because one or more of the applications listed in` *<newresource>* `is not an existing application or its state is incompatible with the state of the controller, or because the list contains duplicate elements.`

  This message appears when the user tries to change the `Resource` attribute of the controller object <*controller*> from <*oldresource*> to <*newresource*> because one or more of the applications listed in <*newresource*> is not an existing application or its state is incompatible with the state of the controller, or because the list contains duplicate elements.

  Action:

  Make sure that the applications listed in the resource <*newresource*> are not written more than once or invalid.

- `(ADM, 100) 74. Dynamic modification failed: because a controller` *<controller>* `has AutoRecover set to 1, its controlled application` *<appname>* `cannot have PreserveState set to 0 or AutoSwitchOver set to 1.`

  If an application needs to be controlled by a controller then the applications' attributes `PreserveState` and `AutoSwitchOver` need to be 1 and 0 respectively if the controller has its AutoRecover set to 1.

  Action:

  Check the PreserveState and `AutoSwitchOver` attribute of the application.

- `(ADM, 106) The total number of SysNodes specified in the configuration for this cluster is` *hosts*`. This exceeds the maximum allowable number of SysNodes in a cluster which is` *maxhosts*`.`

  The total number of `SysNode` objects in the cluster has exceeded the maximum allowable limit.

  Action:

  Make sure that the total number of `SysNode` objects in the cluster does not exceed *maxhosts*.

- `(ADM, 107) The cumulative length of the SysNode names specified in the configuration for the userApplication` <*appname*> `is` *length*`. This exceeds the maximum allowable length which is` *maxlength*`.`

  The cumulative length of the `SysNode` names specified in the configuration for application *appname* exceeds the maximum allowable limit.

  Action:
  Limit the length of the `SysNode` names so that they fit within the maximum allowable limit.

8.3 BAS: Startup and configuration errors

- `(BAS, 2) Duplicate line in hvgdstartup.`

  If RMS detects that a line has been duplicated in the `hvgdstartup`, it prints this error message. The end result of this is that RMS will exit with exit code 23.

  Action:
  Only unique lines are allowed in `hvgdstartup`. Remove all the duplicate entries.

- `(BAS, 3) No kind specified in hvgdstartup.`

  In the `hvgdstartup` file, the entry for the detector is not of the form '`gN -t`<*n*> `-k`<*n*>', or the `-k`<*n*>& option is missing. Since RMS is unable to start, it exits with exit code 23.

  Action:
  Modify the entry for the detector so that the kind (`-k`<*n*> option) for the detector is specified properly.

- `(BAS, 6) DetectorStartScript for kind` <*kind*> `cannot be redefined while detector is running.`

  During dynamic modification, if there is an attempt to redefine the kind for the `DetectorStartScript`, this message is the result.

  Action:
  Do not attempt to redefine the `DetectorStartScript` when the detector is already running.

- (BAS, 9) ERROR IN CONFIGURATION FILE: *message*.

  The *message* can be any one of the following:
  - Check for SanityCheckErrorPrint
  - Object <*object*> cannot have its HostName attribute set since it is not a child of any userApplication. Only the direct descendants of userApplication can have the HostName attribute set.
  - In basic.C:parentsCount(...)
  - The node <*node*> belongs to more than one userApplication, *app1* and *app2*. Nodes must be children of one and only one userApplication node.
  - The node <*node*> is a leaf node and this type <*type*> does not have a detector. Leaf nodes must have detectors.
  - The node <*node*> has an empty DeviceName attribute. This node uses a detector and therefore it needs a valid DeviceName attribute.
  - The rName is <*rname*>, its length *length* is larger than max length *maxlength*.
  - The DuplicateLineInHvgdstartup is <*number*>, so the hvgdstartup file has a duplicate line.
  - The NoKindSpecifiedForGdet is <*number*>, so no kind specified in hvgdstartup.
  - Failed to load a detector of kind <*kind*>.
  - The node <*node*> has an invalid rKind attribute. Nodes of type gResource must have a valid rKind attribute.
  - The node <*node*> has a ScriptTimeout value that is less than its detector report time. This will cause a script timeout error to be reported before the detector can report the state of the resource. Increase the **ScriptTimeout** value for *objectname* (currently *value* seconds) to be greater than the detector cycle time (currently *value* seconds).
  - Node <*node*> has no detector while all its children's "MonitorOnly" attributes are set to 1.
  - The node <*node*> has both attributes "LieOffline" and "ClusterExclusive" set. These attributes are incompatible; only one of them may be used.
  - The type of object <*object*> cannot be or and and at the same time.
  - Object <*object*> is of type and, its state is online, but not all children are online.

Action:
Verify the above description and change the configuration appropriately.

- `(BAS, 14) ERROR IN CONFIGURATION FILE: The object` <*object*> `belongs to more than one userApplication,` *userapplication1* `and` *userapplication2*`. Objects must be children of one and only one userApplication object.`

  An object was encountered as a part of more than one user applications. RMS applications cannot have common objects.

  Action:
  Redesign your configuration so that no two applications have common objects.

- `(BAS, 15) ERROR IN CONFIGURATION FILE: The object` <*object*> `is a leaf object and this type` <*type*> `does not have a detector. Leaf objects must have detectors.`

  An object that has no children objects (i.e. a leaf object) is of type *type* that has no detectors in RMS. All leaf objects in RMS configurations must have detectors.

  Action:
  Redesign your configuration so that all leaf objects have detectors.

- `(BAS, 16) ERROR IN CONFIGURATION FILE: The object` *object* `has an empty DeviceName attribute. This object uses a detector and therefore it needs a valid DeviceName attribute.`

  Critical internal error. If this message appears in switchlog, it indicates a severe problem in the base monitor.

  Action:
  Contact field support.

- `(BAS, 17) ERROR IN CONFIGURATION FILE: The rName is` <*rname*>`, its length` *length* `is larger than max length` *maxlength*`.`

  The value of the `rName` attribute exceeds the maximum length of *maxlength* characters.

  Action:
  Specify shorter `rName`.

- `(BAS, 18) ERROR IN CONFIGURATION FILE: The duplicate line number is` <*linenumber*>`.`

  this message prints out a line number of the duplicate line in `hvgdstartup` file.

  Action:
  Make sure that file `hvgdstartup` has no duplicate lines.

- `(BAS, 19) ERROR IN CONFIGURATION FILE: The NoKindSpecified-ForGdet is` <*kind*>`, so no kind specified in hvgdstartup.`

  The kind has not been specified for the generic detector in the `hvgdstartup` file.

  Action:
  Specify the kind for the generic detector in `hvgdstartup`.

- `(BAS, 24) ERROR IN CONFIGURATION FILE: The object` ***object*** `has an invalid rKind attribute. Objects of type gResource must have a valid rKind attribute.`

  Object *object* has an invalid `rKind` attribute.

  Action:
  Make sure that the object *object* has a valid `rKind` attribute.

- `(BAS, 25) ERROR IN CONFIGURATION FILE: The object` ***object*** `has a ScriptTimeout value that is less than its detector report time. This will cause a script timeout error to be reported before the detector can report the state of the resource. Increase the ScriptTimeout value for` ***object*** `(currently` ***seconds*** `seconds) to be greater than the detector cycle time (currently` ***detectorcycletime*** `seconds).`

  This message is the result of the `ScriptTimeout` value being less than the detector cycle time. This will cause the resource to appear faulted when being brought Online or Offline.

  Action:
  Make the value of `ScriptTimeout` greater than the detector report time.

- `(BAS, 26) ERROR IN CONFIGURATION FILE: The type of object` <*object*> `cannot be 'or' and 'and' at the same time.`

Each RMS object must be of a type derived from `or` or `and` types, but not both. If this message appears in the switchlog, it indicates of a severe corruption of the RMS executable.

Action:
Contact field support.

- `(BAS, 27) ERROR IN CONFIGURATION FILE: object` *<object>* `is of type 'and', its state is online, but not all children are online.`

  This message may appear during dynamic modification, when the existing configuration is checked before applying the modification. If this message appears, the dynamic modification will not proceed.

  Action:
  Make sure that online objects of type `and` have all their children in online states, only then apply dynamic modification.

- `(BAS, 29) ERROR IN CONFIGURATION FILE: object` *<object>* `cannot have its HostName attribute set since it is not a child of any userApplication.`

  An object that is not a child of a `userApplication` has its `HostName` attribute set. Only children of the `userApplication` object can and must have its `HostName` attribute set.

  Action:
  Eliminate the `HostName` attribute from the definition of the object, or disconnect the `userApplication` object from this object, making this object a child of another, non-`userApplication` object.

- `(BAS, 30) ERROR IN CONFIGURATION FILE: The object` ***object*** `has both attributes "LieOffline" and "ClusterExclusive" set. These attributes are incompatible; only one of them may be used.`

  Both attributes `LieOffline` and `ClusterExclusive` are set for the same RMS object. Only one of them can be set for the same object.

  Action:
  Eliminate one or both settings from the RMS object *object*.

- `(BAS, 31) ERROR IN CONFIGURATION FILE: Failed to load a detector of kind `*<kind>*`.`

  A detector was not able to be started by the RMS base monitor.

  Action:
  Make sure detector executable is present in the right place and has executable privileges.

- `(BAS, 32) ERROR IN CONFIGURATION FILE: Object `*<object>*` has no detector while all its children's <MonitorOnly> attributes are set to 1.`

  An object without a detector has all its children's `MonitorOnly` attributes set to 1. An object without a detector must have at least one child for which `MonitorOnly` is set to 0.

  Action:
  Change the configuration so that each object without a detector has at least one child with its `MonitorOnly` set to 0.

- `(BAS, 36) ERROR IN CONFIGURATION FILE: The object `*object*` has both attributes "MonitorOnly" and "ClusterExclusive" set. These attributes are incompatible; only one of them may be used.`

  Both attributes `MonitorOnly` and `ClusterExclusive` are set for the same RMS object. Only one of them can be set for the same object.

  Action:
  Eliminate one or both settings from the RMS object *object*.

8.4 BM: Base monitor

- `(BM, 8) Failed sending message `*<message>*` to object `*<object>*` on host `*<host>*`.`

  When RMS encounters some problems in transmitting the message <*message*> to some other host in the cluster, it prints this message. This could be due to the fact that the RMS on the other host is down or there might be a network problem.

  Action:

Make sure that the RMS is running on the other hosts in the cluster and also whether there are any network issues.

- `(BM, 13) S4: no symbol for object` <*object*> `in .inp file, line =` *linenumber*`.`

  RMS internal error.

  Action:
  Contact field support.

- `(BM, 14) S6: local queue is empty on read directive in line:` *linenumber*`.`

  RMS internal error.

  Action:
  Contact field support.

- `(BM, 15) S2: destination object` <*object*> `is absent in line:` *linenumber*`.`

  RMS internal error.

  Action:
  Contact field support.

- `(BM, 16) S2: sender object` <*object*> `is absent in line:` *linenumber*`.`

  RMS internal error.

  Action:
  Contact field support.

- `(BM, 17) 53. Dynamic modification failed: line` *linenumber*`, cannot build an object of unknown type` <*symbol*>`.`

  An object of unknown type is added during dynamic modification.

  Action:
  Use only objects of known types in configuration files.

- `(BM, 18) 54. Dynamic modification failed: line` *linenumber*`, cannot set value for attribute <`*attribute*`> since object <`*object*`> does not exist.`

  An attribute of a non-existing object cannot be modified.

  Action:
  Modify attributes only for existing objects.

- `(BM, 19) 39. Dynamic modification failed: line` *linenumber*`, cannot modify attribute <`*attribute*`> of object <`*object*`> with value <`*value*`>.`

  Invalid attribute is specified for modification.

  Action:
  Modify only valid attributes.

- `(BM, 20) 77. Dynamic modification failed: line` *linenumber*`, cannot build object <`*object*`> because its type <`*symbol*`> is not a user type.`

  An object <*object*> of a system type <*symbol*> is specified during dynamic modification.

  Action:
  Use only valid resource types when adding new objects to configuration.

- `(BM, 21) 78. Dynamic modification failed: cannot delete object <`*object*`> because its type <`*symbol*`> is not a user type.`

  An object <*object*> of a system type <*symbol*> is specified for deletion.

  Action:
  Delete only objects that are valid resource types.

- `(BM, 23) 80. Dynamic modification failed: The <Follow> attribute for controller <`*controller*`> is set to 1, but the content of a PriorityList of the controlled application <`*controlleduserapplication*`> is different from the content of the PriorityList of the application <`*appname*`> to which <`*controller*`> belongs.`

This message appears when the PriorityList of the controlled application <*controlleduserapplication*> is different from the content of the PriorityList of the application <*appname*> to which the controller <*controller*> belongs.

Action:
Make sure that the PriorityList of the controller and the controlled application is same.

- `(BM, 24) 81. Dynamic modification failed: some resource(s) supposed to come standby failed.`

  During dynamic modification when new resource(s) that are to be added to a resource that is Standby cannot be brought Standby, this message is the result.

  Action:
  Analyze your configuration to make sure that standby capable resources can get to the standby state.

- `(BM, 25) 82. Dynamic modification failed: standby capable controller` <*controller*> `cannot control application` <*appname*> `which has no standby capable resources on host` <*sysnode*>.

  In order for an application <*appname*> to be controlled by a controller <*controller*> the application <*appname*> has to have at least one standby capable resource on host <*sysnode*>.

  Action:
  Make sure that the controlled application has at least one standby capable controller or make sure that the controllers are not standby capable.

- `(BM, 26) 83. Dynamic modification failed: controller` <*controller*> `cannot have attributes StandbyCapable and IgnoreStandbyRequest both set to 0.`

  This message appears when user sets both controller attributes `StandbyCapable` and `IgnoreStandbyRequest` to 1.

  Action:
  Make sure that only one is set to 1 and other to 0.

- `(BM, 29) 84. Dynamic modification failed: controller object` *<controller>* `cannot have its attribute 'Follow' set to 1 while one of OnlineTimeout or StandbyTimeout is not null.`

  The controller node <*controller*> should have one of its attributes Online-Timeout or StandbyTimeout be null to allow the attribute Follow to be 1.

  Action:
  Set the attributes accordingly and try again.

- `(BM, 42) 87. Dynamic modification failed: application` *<appname>* `is not controlled by any controller, but has one of its attributes ControlledSwitch or ControlledShutdown set to 1.`

  This message appears when the user wants the application <*appname*> to be controlled by a controller but one or more of the applications' attributes `ControlledSwitch` or `ControlledShutdown` is set to 1.

  Action:
  Set the attributes accordingly and try again.

- `(BM, 46) 89. Dynamic modification failed: cannot modify a global attribute` *<attribute>* `locally on host` *<hostname>*`.`

  The user cannot modify global attributes <*attribute*> like `DetectorStartScript` or `NullDetector` or `NonCritical` locally on a host <*hostname*>.

  Action:
  Modify the *attribute* globally or modify locally a different attribute.

- `(BM, 54) The RMS-CF-CIP mapping cannot be determined for any host due to the CIP configuration file` *<configfilename>* `missing entries. Please verify all entries in` *<configfilename>* `are correct and that CF and CIP are fully configured.`

  CIP configuration file has missing entries.

  Action:
  Make sure that the CIP configuration has entries for all the RMS hosts that are running in a cluster.

- `(BM, 59) Error` *errno* `while reading line` *<linenumber>* `of .dob file --` *<errorreason>*`.`

During dynamic modification, the base monitor reads its configuration from a '.dob' file. When this file cannot be read, this message appears in the switchlog. The specific OS error is indicated in *errno* and *errorreason*.

Action:

Make sure the host conditions are such that .dob file can be read without errors.

- `(BM, 68) Cannot get message queue parameters using sysdef, errno =` <*errno*>`, reason:` <*reason*>`.`

  While obtaining message queue parameters, `sysdef` was not able to communicate them back to the base monitor. The values of *errno* and *reason* indicate the kind of error.

  Action:

  Contact field support.

- `(BM, 71) 90. Dynamic modification failed: Controller` <*controller*> `has its attribute Follow set to 1. Therefore, its attribute IndependentSwitch must be set to 0, and its controlled application` <*application*> `must have attributes AutoSwitchOver = "No", ControlledSwitch = 1, and Controlled-Shutdown = 1. However, the real values are IndependentSwitch =` <*isw*>`, AutoSwitchOver =` <*asw*>`, ControlledSwitch =` <*csw*>`, and ControlledShutdown =` <*css*>`.`

  When the controller's attribute Follow is set, other attributes such as `IndependentSwitchOver`, `AutoSwitchOver`, `ControlledSwitch`, and `ControlledShutdown` must have the values 0, No, 1, and 1 respectively. However, this condition is violated in the configuration file.

  Action:

  Supply a valid combination of attributes for the controller and its controlled user application.

- `(BM, 72) 91. Dynamic modification failed: Controller` <*controller*> `with the <Follow> attribute set to 1 belongs to an application` <*application*> `which PersistentFault is` <*appfault*>`, while its controlled application` <*controlledapplication*> `has its PersistentFault` <*_fault*>`.`

If controller has its Follow set to 1 then all its controlled applications must have the same value for the attribute `PersistentFault` as the application where the controller belongs to.

Action:
Check and correct the configuration.

- `(BM, 73) The RMS-CF interface is inconsistent and will require operator intervention. The routine "`*routine*`" failed with error code `*errorcode*` - "`*errorreason*`".`

  This is a generic message indicating that the execution of the routine *routine* failed due to the reason *errorreason* and hence the RMS-CF interface is inconsistent. Depending on which routine *routine* has failed, the base monitor can exit with any one of the exit codes 132, 133, 134, 135, 136, 137, 138 or 95.

  Action:
  Contact field support.

- `(BM, 74) The attribute DetectorStartScript and hvgdstartup file cannot be used together. The hvgdstartup file is for backward compatibility only and support for it may be withdrawn in future releases. Therefore it is recommended that only the attribute DetectorStartScript be used for setting new configurations.`

  The attribute `DetectorStartScript` and the file `hvgdstartup` are mutually exclusive.

  Action:
  Make sure that the `DetectorStartScript` be used for setting new configurations as support for `hvgdstartup` may be discontinued in future releases.

- `(BM, 75) 88. Dynamic modification failed: controller `*<controller>*` has its attributes SplitRequest, IgnoreOnlineRequest, and IgnoreOfflineRequest set to 1. If SplitRequest is set to 1, then at least one of IgnoreOfflineRequest or IgnoreOnlineRequest must be set to 0.`

Invalid combination of controller attributes is encountered. If both `IgnoreOfflineRequest` and `IgnoreOnlineRequest` are set to 1, then no request will be propagated to the controlled application(s), so no request can be split.

Action:
Provide a valid combination of the controller attributes.

- `(BM, 80) 92. Dynamic modification failed: controller` *<controller>* `belongs to the application` *<application>* `which AutoSwitchOver attribute has "ShutDown" option set, but its controlled application` *<controlled>* `has not.`

  If a controlling application has its `AutoSwitchOver` attribute set with the option "Shutdown", then all applications controlled by the controllers that belong to this controlling application must also have their `AutoSwitchOver` attributes having the option "Shutdown" set as well.

  Action:
  Provide correct settings for the `AutoSwitchOver` attributes.

- `(BM, 81) 93. Dynamic modification failed: local controller attributes such as NullDetector or MonitorOnly cannot be modified during local modification (hvmod -l).`

  The reason for this message is that the modification of local controller attributes such as `NullDetector` or `MonitorOnly` are allowed only during global modification.

  Action:
  Make a non-local modification, or modify different attributes.

- `(BM, 90) 94. Dynamic modification failed: The length of object name` *<object>* `is` *length*`. This is greater than the maximum allowable length name of` *maxlength*`.`

  The length of object name is greater than the maximum allowable length.

  Action:
  Ensure that the length of the object name is smaller than *maxlength*.

- `(BM, 92) 95. Dynamic modification failed: a non-empty value` *<value>* `is set to <ApplicationSequence> attribute of a non-scalable controller` *<controller>*`.`

A non-scalable controller cannot have its `ApplicationSequence` attribute set to a non-empty value.

Action:

Provide correct settings for the `ApplicationSequence` and `Scalable` attributes.

- `(BM, 94) 97. Dynamic modification failed: the Application-Sequence attribute of a scalable controller` <*controller*> `includes application name` <*hostname*>`, but this name is absent from the list of controlled applications set to the value of` <*resource*> `in the attribute <Resource>.`

  The `ApplicationSequence` attribute of a scalable controller includes an application name absent from the list of the controlled applications.

  Action:

  Provide correct settings for `ApplicationSequence` and `Resource` attributes of the controller.

- `(BM, 96) 94. Dynamic modification failed: a scalable controller` <*controller*> `has its attributes <Follow> set to 1 or <IndependentSwitch> set to 0.`

  A scalable controller must have its attribute `Follow` set to 0 and <IndependentSwitch> set to 1.

  Action:

  Provide correct settings for the `Follow`, `IndependentSwitch`, and `Scalable` attributes.

- `(BM, 97) 95. Dynamic modification failed: controller` <*controller*> `attribute <ApplicationSequence> is set to` <*applica-tionsequence*> `which refers to application(s) not present in the configuration.`

  A scalable controller must list only existing applications in its `Applica-tionSequence` attribute.

  Action:

  Provide correct settings for attribute `ApplicationSequence`.

- `(BM, 98) 96. Dynamic modification failed: two scalable controllers` *<controller1>* `and` *<controller2>* `control the same application` *<application>*`.`

  Only one scalable controller can control an application.

  Action:
  Fix RMS configuration.

- `(BM, 99) 97. Dynamic modification failed: controlled application` *<controlledapp>* `runs on host` *<hostname>*`, but it is controlled by a scalable controller` *<scontroller>* `which belongs to an application` *<controllingapp>* `that does not run on that host.`

  Hostname mismatch between controlled and controlling applications. Controlling application must run on all the hosts where the controlled applications are running.

  Action:
  Fix RMS configuration.

- `(BM, 101) 99. Dynamic modification failed: controlled application` *<controlledapp>* `runs on host` *<hostname>*`, but it is controlled by a scalable controller` *<scontroller>* `which belongs to a controlling application` *<controllingapp>* `that does not allow for the controller to run on that host.`

  Hostname mismatch between controlled and controlling applications. Controlling application must run on all the hosts where the controlled applications are running.

  Action:
  Fix RMS configuration.

- `(BM, 105) 100. Dynamic modification failed: Invalid kind of generic resource specified in DetectorStartScript` *<script>* `for object` *<object>*`.`

  Wrong value is supplied for a flag `-k` in the detector startup script.

  Action:
  Fix RMS configuration.

- `(BM, 106) The rKind attribute of object` *<object>* `does not match the value of the '-k' flag of its associated detector.`

  Values for `rKind` attribute and flag `-k` of the detector startup line do not match.

  Action:
  Fix RMS configuration.

- `(BM, 107) Illegal different values for rKind attribute in object` *<object>*`.`

  Different values for `rKind` attribute are encountered within the same object.

  Action:
  Fix RMS configuration.

- `(BM, 108) 101. Dynamic modification failed: Scalable controller` *<object>* `cannot have its attribute <SplitRequest> set to 1.`

  Setting controller attributes Scalable and `SplitRequest` is mutually exclusive.

  Action:
  Fix RMS configuration.

- `(BM, 109) 102. Dynamic modification failed: Application` *<application>* `has its attribute PartialCluster set to 1 or is controlled, directly or indirectly, via a Follow controller that belongs to another application that has its attribute PartialCluster set to 1 -- this application` *<application>* `cannot have a cluster exclusive resource` *<resource>*`.`

  An exclusive resource cannot belong to an application with the attribute PartialCluster set to 1, or cannot be controlled, directly or indirectly, by a Follow controller from an application with the attribute PartialCluster set to 1.

  Action:
  Fix RMS configuration.

- `(BM, 110) 103. Dynamic modification failed: Application` *<application>* `is controlled by a scalable controller` *<controller>*`, therefore it cannot have its attribute <ControlledShutdown> set to 1 while its attribute <AutoSwitchOver> includes option <ShutDown>.`

  An application controlled by a scalable controller cannot have `ControlledShutdown` set to 1 and `AutoSwitchOver` including the option <ShutDown> at the same time.

  Action:
  Correct RMS configuration.

- `(BM, 111) 104. Dynamic modification failed: Line` *#line* `is too big.`

  A line in a configuration file is too big.

  Action:
  Fix RMS configuration, so that each line takes less than 2000 bytes.

8.5 CML: Command line

- `(CML, 11) Option (`*option*`) requires an operand.`

  Certain options for `hvcm` require an argument. If `hvcm` has been invoked without the argument, this message appears along with the usage and RMS exits with exit code 3.

  Action:
  Check the `hvcm` man page for correct usage.

- `(CML, 12) Unrecognized option` *option*`.`

  The option provided is not a valid one.

  Action:
  Check the `hvcm` man page for correct usage.

- `(CML, 17) Incorrect range argument with -l option.`

  The number for the `-l` option is not correct. Check the range.

  Action:

Check the man page for `hvcm` for range argument with `-l` option.

- `(CML, 18) Log level <`*loglevel*`> is too large. The valid range is 1..`*maxloglevel* `with the -l option.`

  If the loglevel *loglevel* specified with `-l` option for `hvcm` is greater than the maximum possible loglevel *maxloglevel*, this message is the result and RMS exits with exit code 4.

  Action:
  Specify a loglevel between 1 and *maxloglevel* for '`hvcm -l`'.

- `(CML, 19) Invalid range <`*low* – *high*`>. Within the '-l' option, the end range value must be larger than the first one.`

  When a range of loglevels has been specified with `-l` option for `hvcm`, if the value of the end range *high* is smaller than the value of *low*, this message appears and RMS exits with exit code 4.

  Action:
  Specify the end range value to be higher than the initial end range value.

- `(CML, 20) Log level must be numeric.`

  If the log level specified with the `-l` option for `hvcm` is not a number, this message is the result and RMS exits with exit code 4.

  Action:
  Specify a numeric value for the log level.

- `(CML, 21) 0 is an invalid range value. 0 implies all values. If a range is desired, the valid range is 1..`*maxloglevel* `with the -l option.`

  If the log level specified with the `-l` option of `hvcm` is outside the valid range, this message is printed and RMS exits with exit code 4.

  Action:
  The valid range for the `-l` option of `hvcm` is 1..*maxloglevel*.

8.6 CRT: Contracts and contract jobs

- `(CRT, 1) FindNextHost: local host not found in priority list of` *nodename*`.`

  The RMS base monitor maintains a priority list of all the hosts in the cluster. Under normal circumstances, the local host should always be present in the list. If this is not the case, this message is the result.

  Action:
  Contact field support.

- `(CRT, 2) cannot obtain the NET_SEND_Q queue.`

  RMS uses internal queues for sending contracts (contracts are messages that are transmitted between the hosts in a cluster and which ensure that the different hosts are synchronized with respect to a particular operation), be it between processes on the same host or processes on different hosts. If there is a problem with the queue NET_SEND_Q that is being used to transmit these contracts from one host to the other in the RMS cluster, it manifests itself as this message in the switchlog.

  Action:
  Contact field support.

- `(CRT, 3) Message send failed.`

  When RMS tries to send a message to another host in the cluster, if the delivery of this message over the queue NET_SEND_Q has failed, this message is the result. This could be due to the fact that the host that is to receive the message has gone down or there is a problem with the cluster interconnect.

  Action:
  Check to make sure that the other hosts in the cluster are all alive and make sure that none of them are experiencing any network problems.

- `(CRT, 4) Contract retransmit failed: Message Id =` *messageid* `see bmlog for contract details.`

  When RMS on one host sends a contract to another host or itself (if there is only one host in the cluster) over the queue NET_SEND_Q, it tries to transmit this contract a certain number of times which is determined internally. If this message transmission fails even after all these attempts, this message is printed to the switchlog and this contract is discarded (UAP contract is not discarded).

Action:

Make sure that there is no problem with the cluster interconnect. If contract retransmissions occur for `userApplication` contracts, make sure that the cluster is in consistent condition (i.e. no `userApplication` is online on more than one host, no SysNode is in a pending wait state, etc.).

- `(CRT, 5) The contract` <*crtname*> `is being dropped because the local host` <*crthost*> `has found the host originator` <*otherhost*> `in state` <*state*>`. That host is expected to be in state Online. Please check the interhost communication channels and make sure that these hosts see each other Online.`

  The local host *crthost* sees the contract host originator in state *state* when it is expected to be in state Online.

  Action:

  Make sure that the interhost communication channels are working correctly and that the hosts see each other online.

8.7 CTL: Controllers

- `(CTL, 1) Controller` <*controller*> `will not operate properly since its controlled resource` <*resource*> `is not in the configuration.`

  This message appears when a resource is not in the configuration that is controlled by a controller and the controller's `NullDetector` attribute is set to `off`.

  Action:

  The controlled resource must be present in the configuration for the controller to work properly.

- `(CTL, 2) Controller` <*controller*> `detected more than one controlled application Online. This has lead to the controller fault. Therefore, all the online controlled application will now be switched offline.`

If the controller *controller* has two or more of the controlled applications Online on one or more hosts, then the controller faults.

Action:

Make sure that more than one controlled application for a controller is not Online.

8.8 CUP: userApplication contracts

- `(CUP, 2)` *object*`: cluster is in inconsistent condition current online host conflict, received:` *host*`, local:` *onlinenode*`.`

  If the cluster hosts are unable to reach an agreement as to which host is responsible for a particular `userApplication`. The most likely reason for this is an erroneous system administrator intervention (e.g. a forced `hvswitch` request) the `userApplication` is Online on more than one host simultaneously.

  Action:

  Analyze the cluster inconsistency and perform the appropriate action to resolve it. If the application is online on more than one host, shut down ('`hvutil -f`') the `userApplication` on all but one host.

- `(CUP, 3)` *object* `is already waiting for an event cannot set timer!`

  Critical internal error.

  Action:

  Contact field support.

- `(CUP, 5)` *object* `received unknown contract.`

  The contract received by the node from the application is not recognizable. Critical internal error.

  Action:

  Contact field support.

- `(CUP, 7)` *appname* `is locally online, but is also online on another host.`

The user application is already online on other host and is also online in current host.

Action:
User application can only be online on one host. Make sure the application is offline on all but one of the hosts. If this is not the case use '`hvutil -f`' to bring the userApplication to an Offline state on the superfluous hosts.

- `(CUP, 8)` *object*`: could not get an agreement about the current online host; cluster may be in an inconsistent condition!`

  If the cluster hosts are unable to reach an agreement as to which host is responsible for a particular userApplication. The most likely reason for this is, that due to an erroneous system administrator intervention (e.g. a forced `hvswitch` request) the userApplication is Online on more than one host simultaneously.
  Note: This message corresponds to (CUP, 2). While (CUP, 8) is printed on the contract originator, (CUP, 2) is printed on the non-originator hosts.

  Action:
  Analyze the cluster inconsistency and perform the appropriate action to resolve it. If the application is online on more than one host, shut down ('`hvutil -f`') the `userApplication` on all but one host.

8.9 DET: Detectors

- `(DET, 1) FAULT REASON: Resource` <*resource*> `transitioned to a Faulted state due to a child fault.`

  This message appears when the child faulted unexpectedly thereby causing the resource to fault.

  Action:
  Check to see why the child resource has faulted and based on this take corrective action.

- `(DET, 2) FAULT REASON: Resource` <*resource*> `transitioned to a Faulted state due to a detector report.`

  This message is printed when the detector unexpectedly reports Faulted state.

Action:
Check to see why the resource has faulted and take appropriate action.

- `(DET, 3) FAULT REASON: Resource` *<resource>* `transitioned to a Faulted state due to a script failure.`

  This message appears when the detector failed to execute the script for a resource.

  Action:
  Ensure that there is nothing wrong with the script and also check the resource for any problems.

- `(DET, 4) FAULT REASON: Resource` *<resource>* `transitioned to a Faulted state due to a FaultScript failure. This is a double fault.`

  When a resource faults due to some reason, it runs its Fault script, but in this case the Fault script failed to execute for that resource.

  Action:
  Check to see if there is a problem with the resource or with the Fault script.

- `(DET, 5) FAULT REASON: Resource` *<resource>* `transitioned to a Faulted state due to the resource failing to come Offline after running its OfflineScript (`*offlinescript*`).`

  After a resource executes its offline script, it is expected to come Offline. If it does not change its state, or transitions to a state other than Offline within the period of seconds specified by its `ScriptTimeout` attribute, the resource is considered as being Faulted.

  Action:
  Make sure the Offline script moves the resource into Offline state.

- `(DET, 6) FAULT REASON: Resource` *<resource>* `transitioned to a Faulted state due to the resource failing to come Online after running its OnlineScript (`*onlinescript*`).`

  After a resource executes its online script, it is expected to come Online. If it does not change its state, or transitions to a state other than Online within the period of seconds specified by its `ScriptTimeout` attribute, the resource is considered as being Faulted.

Action:
Make sure the Online script moves the resource into Online state.

- `(DET, 7) FAULT REASON: Resource` *<resource>* `transitioned to a Faulted state due to the resource unexpectedly becoming Offline.`

  This message appears when the resource becomes Offline unexpectedly.

  Action:
  Check to see why the resource suddenly transitioned to the Offline state.

- `(DET, 11) DETECTOR STARTUP FAILED: Corrupted command line` *<commandline>*`.`

  Critical internal error. This message occurs when the command line is empty or has some incorrect value.

  Action:
  Contact field support.

- `(DET, 12) DETECTOR STARTUP FAILED` *<detector>*`. REASON:` *error-reason*`.`

  If the detector *detector* could not be started due to *errorreason*, this message is the result. The reason *errorreason* could be any one of the following:
  – The detector *detector* does not exist.
  – The detector *detector* does not have execute permission.
  – The process for the detector could not be spawned.
  – If the number of processes created by the base monitor at the same time is greater than 128.

  Action:
  Depending on what the reason for the error is take appropriate action.

- `(DET, 13) Failed to execute script` *<script>*`.`

  The detector script is not good or the format is not good.

  Action:
  Check the detector startup script.

- `(DET, 24) FAULT REASON: Resource` <*resource*> `transitioned to a Faulted state due to the resource failing to come Standby after running its OnlineScript (`*onlinescript*`).`

  After a resource executes its online script during standby request, it is expected to come Standby. If it does not change its state, or transitions to a state other than Standby or Online within the period of seconds specified by its `ScriptTimeout` attribute, the resource is considered as being Faulted.

  Action:
  Make sure the Online script moves the resource into Standby or Online state during standby request.

- `(DET, 26) FAULT REASON: Resource` <*resource*> `transitioned to a Faulted state due to the resource failing to come Online.`

  This message appears when the resource fails to come Online after executing it Online scripts that may transition the state of the resource to faulted.

  Action:
  Check to see what prevented the resource *resource* from coming Online.

- `(DET, 28)` <*object*>`: CalculateState() was invoked for a non-local object! This must never happen. Check for possible configuration errors!`

  During the processing of a request within the state engine, a "request or response token" was delivered to an object that is not defined for the local host. Critical internal error.

  Action:
  Contact field support.

- `(DET, 33) DETECTOR STARTUP FAILED: Restart count exceeded.`

  When a detector dies, RMS attempts to restart it. If a detector successfully restarts and once again dies too many times within one minute, RMS assumes there is a problem, terminates the restart cycle, and prints this message.

  Action:
  Contact field support.

- `(DET, 34) No heartbeat has been received from the detector with pid` *<pid>*`,` *<startupcommand>*`, during the last` *<seconds>* `seconds. The base monitor will send the process a SIGALRM to interrupt the detector if it is currently stalled waiting for the alarm.`

  In order to avoid stalling of RMS detectors, each detector periodically sends a heartbeat message to the base monitor. When the heartbeat is missing for a period of time, the base monitor prints this message into switchlog. The base monitor will send an alarm signal to the stalled process to ensure the detector will properly handle its main loop responsibilities. If the amount of time stated since the last time the base monitor had received the heartbeat from the detector exceeds 300 seconds, then the message may indicate the base monitor is not allowed to run. Currently, the base monitor is a real-time process, but not locked in memory. This message may also occur because the bm process has been swapped out and has not had a chance to run again.

  Action:
  Make sure that the base monitor and detector are active using system tools such as truss(1) or strace(1). If the loss of heartbeat greatly exceeds the 300 second timeout, then this may require that system swap or main memory is insufficient.

8.10 GEN: Generic detector

- `(GEN, 1) Usage:` *command* `-t time_interval -k <kind> [-d]`

  Usage error for <*command*>.

  Action:
  Use the specified syntax for the command.

- `(GEN, 2) Memory lock failed.`

  Action:
  Critical error. Contact field support.

- `(GEN, 3) Cannot open` *command* `log file.`

  The file <*command*>log used for logging could not be opened.

  Action:

Contact field support.

- `(GEN, 4) failed to create mutex:` *directory*

  The various RMS commands like `hvdisp`, `hvswitch`, `hvutil` and `hvdump` utilize the lock files from the directory <*directory*> for signal handling purposes. These files are deleted after these commands are completed. The locks directory is also cleaned when RMS starts up. If they are not cleaned for some reason, this message is the result. RMS exits with exit code 99.

  Action:
  Make sure that the locks directory <*directory*> exists.

- `(GEN, 5)` *command*`: failed to get information about RMS base monitor bm!`

  The generic detector <*command*> was unable to get any information about the base monitor.

  Action:
  Contact field support.

- `(GEN, 7)` *command*`: failed to lock virtual memory pages, errno =` *value*`, reason:` *reason*`.`

  The generic detector <*command*> was not able to lock its virtual memory pages in physical memory.

  Action:
  Contact field support.

8.11 INI: init script

- `(INI, 1) Cannot open file` *dumpfile*`, errno =` *errno*`:` *explanation*`.`

  This message appears when the file <*dumpfile*> failed to open because of the error code <*errno*>, explained in <*explanation*>.

  Action:
  Correct the problem according to <*explanation*>.

- `(INI, 9) Cannot close file` *dumpfile*`, errno =` *errno*`:` *explanation*`.`

This message appears when the file <*dumpfile*> failed to close because of the error code <*errno*>, explained in <*explanation*>.

Action:
Correct the problem according to <*explanation*>.

8.12 MIS: Miscellaneous

- `(MIS, 1) No space for object.`

Action:
Critical error. Contact field support.

8.13 NOD: Node detector

- `(NOD, 6) Usage:` *detector* `-t time_interval`

If the detector <*detector*> has been provided a non-integer argument, this message is the result and the detector exits with exit code 103.

Action:
Provide an integer as the <*time_interval*> for the detector <*detector*>.

- `(NOD, 7) cluster host` *host* `is no longer in time sync with local node. Sane operation of RMS can no longer be guaranteed. Further out-of-sync messages will appear in the syslog.`

The time on <*host*> is not in sync with the time on the local node.

Action:
Sync the time on <*host*> with the time on the local node.

- `(NOD, 8) Usage:` *detector* `-t time_interval [-d] [-n]`

If the argument '`-t` <*time_interval*>' has not been provided for the detector <*detector*> or if an argument other than `-d` or `-n` is used, this message is printed to the switchlog and the detector exits with exit code 103.

Action:
Use the specified syntax for the invocation of the detector.

- `(NOD, 9)` *detector*`: Failed to open req_queue.`

  The detector `hvdet_node` utilizes the queue `req_queue` for getting jobs from the base monitor. If there is some problem with the queue, this message is the result with the detector exiting with exit code 106.

  Action:
  Contact field support.

- `(NOD, 10)` *detector*`: Failed to open rep_queue.`

  The detector `hvdet_node` utilizes the queue `rep_queue` to report the state of the other `SysNodes` in the cluster to the base monitor running on the same host as the detector. If there is a problem in sending the state over to the base monitor, this message is printed out and the detector exits with exit code 112.

  Action:
  Contact field support.

- `(NOD, 11)` *service*`: getservbyname returned NULL.`

  If the detector has been unable to find the port at which the service <*service*> resides, this message is printed to the switchlog and the detector exits with exit code 126.

  Action:
  This is probably due to the absence of an entry for service <*service*> in `/etc/services`.

- `(NOD, 12)` *detector*`: no NODE_SYS_Q.`

  The detector `hvdet_node` uses the queue `NODE_SYS_Q` to get the list of `SysNodes` from the base monitor running on the same host as the detector. The detector tries to create this queue until it is successful or for 10 times whichever is shorter. If after these attempts it is still unsuccessful it prints out the above message and exits with exit code 106.

  Action:
  Contact field support.

- `(NOD, 13) The RMS-CF-CIP mapping for SysNode` <*sysnode*> `to the CIP name has failed. Please verify all entries in /etc/hosts and /etc/cip.cf are correct and that CF and CIP are fully configured.`

If there is no CIP entry corresponding to the `SysNode` <*sysnode*> in `/etc/cip.cf`, this message is the result and `hvdet_node` exits with exit code 139.

Action:
Make sure that there is a corresponding CIP entry for the `SysNode` <*sysnode*> in `/etc/cip.cf`.

- `(NOD, 16)` *detector*`: failed to get information about RMS base monitor bm!`

  When the detector `hvdet_node` finds that the RMS base monitor is not, it exits with exit code 142.

  Action:
  This might be due to the fact that `hvdet_node` has been started independently of RMS.

- `(NOD, 17) Failed to set up SIGCHLD handler!`

  Action:
  Critical error. Contact field support.

- `(NOD, 18) Can't fork child hvdet_node.`

  Action:
  Critical error. Contact field support.

- `(NOD, 20)` *detector*`: Cannot create socket:` *errorreason*`.`

  If there is a problem in the creation of an endpoint for communication between the detectors (<*detector*>) on the different hosts in the cluster, it manifests itself as a message in the switchlog and the detector exits with the exit code 111.

  Action:
  Contact field support.

- `(NOD, 21)` *detector*`: Failed to bind address to socket:` *error-reason*`.`

  If there is a problem in binding the endpoint of communication between the detectors (<*detector*>) on the different hosts in the cluster to a particular port, the result is this message with <*errorreason*> indicating the reason for this error. The detector then exits with exit code 130.

Action:

Contact field support.

- `(NOD, 22) The interconnect` ***interconnect*** `to the cluster host` ***host*** `failed.`

  Action:

  Critical error. Contact field support.

- `(NOD, 25) The network connection to the cluster host` ***host*** `failed.`

  Action:

  Critical error. Contact field support.

- `(NOD, 26)` ***detector***`: detector can't report resource state.`

  If the detector <*detector*> cannot report the state of the other `SysNodes` in the cluster to the base monitor running on the same host as the detector, this message is the result. This is most likely a problem with the queue when the detector is reporting the state.

  Action:

  Contact field support.

- `(NOD, 28)` ***detector***`: SysNode list empty in hvdet_node.`

  The `hvdet_node` contacts the base monitor to get the list of `SysNodes` but if it just gets an empty list back in return, this message is the result. RMS then exits with exit code 129.

  Action:

  Contact field support.

- `(NOD, 29) The RMS-CF interface is inconsistent and will require operator intervention. The routine "`***routine***`" failed with error code` ***errorcode*** `- "`***errorreason***`".`

  This is a generic message indicating that the execution of the routine <*routine*> failed due to the reason <*errorreason*> and hence the RMS-CF interface is inconsistent. Depending on which routine <*routine*> has failed, the detector `hvdet_node` can exit with any one of the exit codes 132, 133, 134, 135, 136, 137, 138 or 95.

  Action:

Contact field support.

- `(NOD, 30)` *detector*`: message get doesn't work in hvdet_node.`

  When the `hvdet_node` contacts the base monitor to get the list of `SysNodes`, if it finds that it could not get the list of `SysNodes` even after trying 10 times, this message is printed to the switchlog. This means that there is some problem with the message queues between the `hvdet_node` and the base monitor. RMS then exits with exit code 129.

  Action:
  Contact field support.

- `(NOD, 31)` *detector*`: nodename` *nodename* `not in NODELIST.`

  This message indicates a severe malfunction in RMS, when the detector <*detector*> cannot find the node <*nodename*> in its list of nodes.

  Action:
  Contact field support.

- `(NOD, 33) The` *interface* `interface connection to the cluster host` *host* `failed.`

  Action:
  Critical error. Contact field support.

- `(NOD, 34)` *detector*`: Failed to call osd select:` *errorreason*`.`

  If the detector `hvdet_node` fails during the system call select while reading messages, this message is printed to the switchlog along with the reason <*errorreason*>. The detector then exits with exit code 131.

  Action:
  Contact field support.

- `(NOD, 37) Child hvdet_node died. Will try to restart hvdet_node.`

  Action:
  None required.

- `(NOD, 38) cluster host` *host* `is no longer in time sync with local node. Sane operation of RMS can no longer be guaranteed.`

  The time on the cluster host <*host*> differs significantly (> 5 times the `hvdet_node` interval) from the local node.

  Action:
  Make sure that all the cluster hosts are in time sync.

- `(NOD, 40)` *command*`: gethostbyname returned NULL for host` *hostname*`.`

  If there is a problem in the detector when resolving a host <*hostname*>, this message is the result and the detector exits with exit code 114.

  Action:
  Make sure that you provide a valid host name.

8.14 QUE: Message queues

- `(QUE, 13) RCP fail:` *filename* `is being copied.`

  If there is an attempt to copy the file with name *filename* when there is another copy in progress, this message is the result.

  Action:
  Make sure that concurrent copies of the same file do not occur.

- `(QUE, 14) RCP fail: fwrite errno` *errno*`.`

  There was a problem while transferring files from one cluster host to the other.

  Action:
  Take action based on the *errno*.

8.15 SCR: Scripts

- `(SCR, 8) Invalid script termination for controller` <*controller*>`.`

  The controller script is not correct or invalid.

Action:
Check the controller script.

- `(SCR, 9) REASON: failed to execute script <`*script*`> with resource <`*resource*`>:` *errorreason*`.`

  The detector script is not good or the format is not good.

  Action:
  Check the detector script.

- `(SCR, 20) The attempt to shut down the cluster host` *host* `has failed:` *errorreason*`.`

  The cluster host could not be killed because of one of the following reasons:
  – Script exited with a non-zero status.
  – Script exited due to signal caught.
  – Other unknown failure.

  Action:
  Verify the status of the node, make any necessary corrections to the script, potentially correct the node state manually if possible and issue appropriate '`hvutil -{o, u}`' as needed.

- `(SCR, 21) Failed to execute the script <`*script*`>, errno = <`*errno*`>, error reason: <`*errorreason*`>.`

  If the *script* cannot be executed, this message is printed out along with the *errorreason*.

  Action:
  Take action based on the *errorreason*.

8.16 SWT: Switch requests (hvswitch command)

- `(SWT, 4)` *object* `is online locally, but is also online on` *onlinenode*`.`

  If the object *object* is online on more than one host, this message is the result.

  Action:

Make sure that the object *object* is online on only one host in the cluster.

- `(SWT, 20) Could not remove host <`*hostname*`> from local priority list.`

  A host has left the cluster, but RMS was unable to remove the corresponding entry from its internal Priority List. This is an internal problem in the program stack and memory management.

  Action:
  Contact field support.

- `(SWT, 25)` *objectname*`: outstanding switch request of dead host was denied; cluster may be in an inconsistent condition!`

  A host died during the processing of a switch request. If the host that takes over the responsibility for that particular userApplication tried to proceed with the partly-done switch request, but another host does not agree. This indicates a severe cluster inconsistency and critical internal error.

  Action:
  Contact field support.

- `(SWT, 26)` *object*`: dead host <`*hostname*`> was holding an unknown lock. Lock will be skipped!`

  This message appears when the dead host <*hostname*> was holding a lock that is unknown to the new responsible host.

  Action:
  Allow time for the cluster to cleanup.

- `(SWT, 45) hvshut aborted because of a busy uap <`*appname*`>.`

  The `hvshut` request was aborted because the application is busy.

  Action:
  Do not shut down RMS when its applications are busy. Make sure the application finishes its processing before shutting down RMS.

- `(SWT, 46) hvshut aborted because modification is in progress.`

The `hvshut` request was aborted because dynamic modification is in progress.

Action:
Do not shut down RMS while dynamic modification is in progress. Wait until dynamic modification finishes before shutting down RMS.

8.17 SYS: SysNode objects

- `(SYS, 1) Error on SysNode:` *object*`. It failed to send the kill success message to the cluster host:` *host*`.`

  When a cluster host is killed, the host requested the kill must send a success message to the surviving hosts. This message appears in the switchlog when this message send fails.

  Action:
  Make sure the cluster and network conditions are such that the message can be sent across the network.

- `(SYS, 8) RMS failed to shut down the host` *host* `via the Shutdown Facility, no further kill functionality is available. The cluster is now hung.`

  This message appears when the RMS was sending a kill request to the Shutdown Facility and did not get the elimination acknowledgement.

  Action:
  Refer to the manuals of the ShutDown Facility to find out what was going wrong with the host elimination. Check the actual status of the remote host and invoke the appropriate '`hvutil -u`' or '`hvutil -o`' command to resolve the RMS hang state.

- `(SYS, 13) Since this host` <*hostname*> `has been online for no more than` *time* `seconds, and due to the previous error, it will shut down now.`

  This message appears when the checksum of this host is different from the hosts in the cluster (one of the possible reasons).

  Action:
  Check the configuration in all the cluster hosts and verify that same configuration is running on all of them.

- `(SYS, 14) Neither automatic nor manual switchover will be possible on this host until` <*detector*> `detector will report offline or faulted.`

  When different configurations are encountered in a cluster where one host is offline and the other is online.

  Action:
  Run the same configuration in a single cluster or different clusters do not have common hosts.

- `(SYS, 15) The uname() system call returned with Error. RMS will be unable to verify the compliance of the RMS naming convention!`

  This message appears when `uname()` system call returned with a non-zero value.

  Action:
  Make sure that the `SysNode` name is valid and restart RMS as needed.

- `(SYS, 17) The RMS internal SysNode name "`*sysnode*`" is ambiguous with the name "`*name*`". Please adjust names compliant with the RMS naming convention "SysNode = `uname -n`RMS"`

  The RMS naming convention '_sysnodename_ `= `uname -n`RMS`' is intended to allow use of the CF-name with and without trailing "RMS" whenever an RMS command expects a `SysNode` reference. This rule creates an ambiguity if one `SysNode` is named "*xxx*`RMS`" and another is named "*xxx*", because '_rms_command_ *xxx*' could refer to either `SysNode`. Therefore, ambiguous `SysNode` names are not be allowed.

  Action:
  Use non-ambiguous `SysNode` names and adhere to the RMS naming conventions.

- `(SYS, 48) Remote host` <*hostname*> `replied the checksum` <*remotechecksum*> `which is different from the local checksum` <*localchecksum*>`. The sysnode of this host will not be brought online.`

  This message appears when the remote host <*hostname*> is running different configuration than the local host or different loads of RMS package are installed on these hosts.

Action:

Make sure all the hosts are running the same configuration and the configuration is distributed on all hosts. Make sure that same RMS package is installed on all hosts (same load).

- `(SYS, 49) Since this host <`*`hostname`*`> has been online for more than `*`time`*` seconds, and due to the previous error, it will remain online, but neither automatic nor manual switchover will be possible on this host until <`*`detector`*`> detector will report offline or faulted.`

  This message appears when the checksum of this host is different from the hosts in the cluster (one of the possible reasons).

  Action:

  Check the configuration in all the cluster hosts and verify that same configuration is running on all of them.

- `(SYS, 50) Since this host <`*`hostname`*`> has been online for no more than `*`time`*` seconds, and due to the previous error, it will shut down now.`

  This message appears when the checksum of this host is different from the hosts in the cluster (one of the possible reasons).

  Action:

  Check the configuration in all the cluster hosts and verify that same configuration is running on all of them.

- `(SYS, 84) Request <hvshut -a> timed out. RMS will now terminate! Note: some cluster hosts may still be online!`

  This message appears when the default timeout for the `hvshut` command expired and some of the hosts are still running.

  Action:

  Adjust the default timer by setting `RELIANT_SHUT_MIN_WAIT` to a value, which is large enough to allow a shutdown on all hosts. Check if shutdown fails for internal problems (e.g. a failure of an OfflineScript cause an userApplication to fail to go Offline).

- `(SYS, 90) `*`hostname`*` internal WaitList addition failure! Cannot set timer for delayed detector report action!`

System Error.

Action:
Contact field support.

- `(SYS, 93) The cluster host` ***nodename*** `is not in the Wait state. The hvutil command request failed!`

  This message appears when the user issues the `hvutil` command ('`hvutil -o`' or '`hvutil -u`') and the cluster host <*nodename*> is not in the Wait state.

  Action:
  Reissue '`hvutil -{o, u}`' only when the host is in a Wait state.

- `(SYS, 94) The last detector report for the cluster host` ***hostname*** `is not online. The hvutil command request failed!`

  This message appears when the user issues the `hvutil` command ('`hvutil -o` *sysnode*') to clear the Wait state of the `SysNode` and the `SysNode` is still in Wait state because the last detector report for the cluster host <*hostname*> is not Online i.e. the `SysNode` might have transitioned to Wait state not from Online but from some other state.

  Action:
  Issue '`hvutil -o`' only when the host is in a Wait state that has transitioned from the Online state.

- `(SYS, 97) Cannot access the NET_SEND_Q queue.`

  When a new host comes Online, the other hosts in the cluster try to determine if the new host has been started with `-C` option. The host that has just come online uses the queue NET_SEND_Q to send the necessary information to the other hosts in the cluster. If this host is unable to access the queue NET_SEND_Q this message is printed.

  Action:
  Contact field support.

- `(SYS, 98) Message send failed in SendJoinOk.`

  When a new host comes Online, the other hosts in the cluster try to determine if the new host has been started with `-C` option. The host that has just come online uses the queue NET_SEND_Q to send the necessary information to the other hosts in the cluster. If this host is unable to send the necessary information to the other hosts in the cluster, this message is printed.

Action:
Check if there is a problem with the network.

8.18 UAP: userApplication objects

- `(UAP, 1) Request to go online will not be granted for application` <*appname*> `since the host` <*sysnode*> `runs a different RMS configuration.`

  This message appears when the request is done for an application <*appname*> to go Online but the host <*sysnode*> is running a different configuration.

  Action:
  Make sure that the user is running the same configuration.

- `(UAP, 5)` *object*`: cmp_Prio:` *list*`.`

  This message is the result of the priority list *list* having invalid entries.

  Action:
  Contact field support.

- `(UAP, 6) Could not add new entry to priority list.`

  Critical internal error.

  Action:
  Contact field support.

- `(UAP, 7) Could not remove entries from priority list.`

  Critical internal error.

  Action:
  Contact field support.

- `(UAP, 8)` *object*`: cpy_Prio failed, source list corrupted.`

This message appears when either the PriorityList is empty or the list is corrupted. Critical internal error.

Action:
Contact field support.

- `(UAP, 9)` ***object***`: Update of PriorityList failed, cluster may be in inconsistent condition.`

  If a contract that is supposed to be present in the internal list does not exist, this message is the result. The cluster may be in an inconsistent condition.

  Action:
  Contact field support.

- `(UAP, 15)` ***sysnode***`: PrepareStandAloneContract() processing unknown contract.`

  This message appears when there is only one application <*sysnode*> Online and has to process a contract that is not supported. Critical internal error.

  Action:
  Contact field support.

- `(UAP, 16)` ***object***`::SendUAppLockContract: local host doesn't hold a lock -- Contract processing denied.`

  This message appears when the contract is processed by the local host, which does not have the lock for that application contract. Critical internal error.

  Action:
  Contact field support.

- `(UAP, 19)` ***object***`::SendUAppLockContract: LOCK Contract cannot be sent.`

  This message appears when the LOCK contract cannot be sent over the network.

  Action:
  The network may be down.

- `(UAP, 21)` *object*`::SendUAppUnLockContract: UNLOCK Contract cannot be sent.`

  This message appears when the UNLOCK contract cannot be sent over the network.

  Action:
  The network may be down.

- `(UAP, 22)` *object* `unlock processing failed, cluster may be in an inconsistent condition!`

  This message appears when the local node receives a UNLOCK contract but is unable to perform the follow up processing, which was committed in the contract.

  Action:
  Contact field support.

- `(UAP, 23)` *object* `failed to process UNLOCK contract.`

  A host was unable to propagate the received UNLOCK contract, e.g., because of networking problems or memory problems.

  Action:
  This message should appear with an additional ERROR message specifying the origin of the problem. Refer to the ERROR message.

- `(UAP, 24) Deleting of local contractUAP object failed, cannot find object.`

  This message appears when the local contract node has completed the contract and has sent it to the local node but the local node could not able to find it.

  Action:
  Contact field support.

- `(UAP, 27)` *object* `received a DEACT contract in state:` *state*`.`

  The correspondent `userApplication` on a remote host is in the `DeAct` state, but the local `userApplication` is not. Critical internal error.

  Action:
  Contact field support.

- `(UAP, 28)` *object* `failed to update the priority list. Cluster may be in an inconsistent state.`

  When the local host receives a contract for unlocking the hosts in the cluster with respect to a particular operation, if the local host finds that a particular host has died, it updates its priority list to reflect this, but if it is unable to perform this operation due to some reason, this message is the result. This indicates a critical internal problem in memory management.

  Action:
  Contact field support.

- `(UAP, 29)` *object*`: contract data section is corrupted.`

  This message appears when the application is unable to read the data section of the contract.

  Action:
  Contact field support.

- `(UAP, 32)` *object* `received unknown contract.`

  This message appears when the application unable to unlock the contract as it was unable to find the kind of contract request in its code that it expected. Critical internal error.

  Action:
  Contact field support.

- `(UAP, 33)` *object* `unknown task in list of outstanding contracts.`

  This message appears when a userApplication object finds a task in the list of outstanding contracts but unable to process it as it could not able to find the kind of contract request in its code. Critical internal error.

  Action:
  Contact field support.

- `(UAP, 35)` *object*`: inconsistency occurred. Any further switch request will be denied (except forced requests). Clear inconsistency before invoking further actions!`

  This message appears when the state of the application is offline or standby and some of the resources are online and faulted.

Action:
Clear the inconsistency by the appropriate command (usually '`hvutil -c`').

- `(UAP, 41) cannot open file` ***filename***`. Last Online Host for userApplication cannot be stored into non-volatile device.`

  File open error.

  Action:
  Check the reliant path.

- `(UAP, 42) found incorrect entry in status file:` *>entry<*

  This message appears when the status_info file has incorrect entry in it. This should occur only if the status info file was edited manually.

  Action:
  Check the status info file for manual incorrect entries. If this is not the case contact field support.

- `(UAP, 43)` *<object>*`: could not insert` *<host>* `into local priority list.`

  Action:
  Critical error. Contact field support.

- `(UAP, 44)` *<object>*`: could not remove` *<host>* `from local priority list.`

  Action:
  Critical error. Contact field support.

- `(UAP, 45)` *<object>*`: could not remove` *<host>* `from priority list.`

  Action:
  Critical error. Contact field support.

8.19 US: us files

- `(US, 5) The cluster host` ***hostname*** `is no longer reachable! Please check the status of the host and the ethernet connection.`

This message is a result of one cluster host detecting that the other host *hostname* which is part of the cluster is no longer reachable or in other words this cluster host sees the other host *hostname* as faulted. This could be due to the fact that the other host *hostname* has gone down or there is some problem with the cluster interconnect.

Action:
Check if the host *hostname* is indeed dead, if not check if there is a problem with the ethernet connection.

- `(US, 6) RMS has died unexpectedly on the cluster host` *hostname*`!`

When the detector on the local host detects that the host *hostname* has transitioned from Online to Offline unexpectedly, it prints this message to the switchlog and then it attempts to kill the host *hostname*.

Action:
Check the syslog on the host *hostname* to find out the reason why it has gone down.

- `(US, 31) FAULT REASON: Resource` ***resource*** `transitioned to a Faulted state due to a detector report.`

This message is printed when the detector unexpectedly reports Faulted state.

Action:
Check to see if there is any problem with the *resource*.

8.20 WLT: Wait list

- `(WLT, 1) REASON: Resource` ***resource***`'s` ***script*** `(`***scriptexecd***`) has exceeded the ScriptTimeout of` ***timeout*** `seconds.`

The detector script for the resource has exceeded the `ScriptTimeout` limit.

Action:
Make sure that timeout is large enough to execute the script.

- `(WLT, 3) Cluster host` *hostname*`'s Shutdown Facility invoked via (`*script*`) has not finished in the last` *time* `seconds. An operator intervention is required!`

  The Shutdown Facility that is killing host *hostname* has not terminated yet. Operator intervention may be required. This message will appear periodically (with the period equal to the node's `ScriptTimeout` value), until either the script terminates on its own, or until the script is terminated by the Unix `kill` command. If terminated by the `kill` command, the host being killed will not be considered killed.

  Action:
  Wait until the script terminates, or terminate the script using `kill` command if the script cannot terminate on its own.

- `(WLT, 5) CONTROLLER FAULT: Controller` *<object>* `has propagated` *<request>* `request to its controlled application(s)` *<applications>*`, but the request has not been completed within the period of` *<timeout>* `seconds.`

  When controller propagates its requests to the controlled applications, it is waiting for the completion of the request for a period of time sufficient for the controlled applications to process the request. When the request if not completed within this period, controller faults.

  Action:
  Fix the controller's scripts and/or scripts of the controlled applications, or repair resources of the controlled applications. For user defined controller scripts increase their `ScriptTimeout` values.

8.21 WRP: Wrappers

- `(WRP, 1) Failed to set script to TS.`

  The script could not be made into a time sharing process.

  Action:
  Take action based on the reason.

- `(WRP, 2) Illegal flag for process wrapper creation.`

  Action:
  Critical error. Contact field support.

- `(WRP, 3) Failed to execv:` *command*`.`

  This message could occur in any of the following scenarios:
  – A detector cannot be started because RMS is unable to create the detector process with the command *command*.
  – '`hvcm -a`' has been invoked and the RMS base monitor cannot be started on the individual hosts comprising the cluster with the command *command*.
  – A script cannot be started because RMS is unable to create the script process with the command *command*.
  RMS shuts down on the node where this message appears and returns an error number *errno*, which is the error number returned by the operating system.

  Action:
  Consult the system manual pages or the appendix of this manual for the explanation for error number *errno* and see if the cause is evident. If not, contact field support.

- `(WRP, 4) Failed to create a process:` *command*`.`

  This message could occur in any of the following scenarios:
  – A detector cannot be started because RMS is unable to create the detector process to execute the command *command*.
  – '`hvcm -a`' has been invoked and the RMS base monitor cannot be started on the individual hosts comprising the cluster with the command *command*.
  – A script cannot be started because RMS is unable to create the script process with the command *command*.
  RMS shuts down on the node where this message appears and returns an error number *errno*, which is the error number returned by the operating system.

  Action:
  Consult the system manual pages or the appendix of this manual for the explanation for error number *errno* and see if the cause is evident. If not, contact field support.

- `(WRP, 5) No handler for this signal event` <*signal*>`.`

  There is no signal handler associated with the signal *signal*.

  Action:
  Contact field support.

- `(WRP, 6) Cannot find process (pid=`*processid*`) in the process wrappers.`

  Action:
  Critical error. Contact field support.

- `(WRP, 7) getservbyname failed for service name:` *servicename*`.`

  Action:
  Critical error. Contact field support.

- `(WRP, 8) gethostbyname failed for remote host:` *host*`.`

  Action:
  Critical error. Contact field support.

- `(WRP, 9) Socket open failed.`

  This message occurs if RMS is unable to create a datagram endpoint for communication.

  Action:
  Contact your System Administrator.

- `(WRP, 10) connect to server failed.`

  Action:
  Critical error. Contact field support.

- `(WRP, 11) Message send failed, queue id <`*queueid*`>, process <`*process*`>, <`*name*`>, to host <`*host*`>.`

  RMS exchanges messages between processes and hosts to maintain inter-host communication. If the delivery of a message has failed then this error is the result. This can occur if one or more hosts in the cluster are not active or if there is a problem with the network.

  Action:
  (i) Check the other hosts in the cluster. If any are not alive, check the switchlog for information regarding why RMS has died on those hosts. Perform the following steps in order:
  1. '`hvdisp -a`'
  2. In the output of step (a) check if the state of any of the resources whose type is `SysNode` is offline. If so, that means that RMS is not running on that node.

3. Check the switchlogs of all the nodes that are offline to determine the reason why RMS on that node is not active.
(ii) If the other hosts that are part of the cluster are alive then that means there is some problem with the network.

- `(WRP, 12) Failed to bind port to socket.`

  This could occur if RMS is unable to bind the endpoint for communication.

  Action:
  Contact field support.

- `(WRP, 14) No available slot to create a new host instance.`

  When the base monitor for RMS starts up, it creates a slot in an internal data structure for every host in the cluster.
  When `hvdet_node` is started up, RMS sends it a list of the `SysNode` objects that are put into different slots in the internal data structure. If the data structure has run out of slots (16) to put the `SysNode` name in, this message is printed out.

  Action:
  Contact field support.

- `(WRP, 15) gethostbyname(`*hostname*`): host name should be in /etc/hosts`

  When the hostname *hostname* specified as a `SysNode` does not have an entry in `/etc/hosts`, this message is printed out to the switchlog.

  Action:
  Correct the host name *hostname* to be an entry in `/etc/hosts`.

- `(WRP, 16) No available slot for host` *hostname*

  When RMS has run out of slots for the cluster interfaces (64), this message is printed along with the host name *hostname* for which this happened.

  Action:
  Contact field support.

- `(WRP, 17) Size of integer or IP address is not 4-bytes`

Critical internal error.

Action:
Contact field support.

- `(WRP, 18) Not enough memory in` *<processinfo>*

  Action:
  Critical error. Contact field support.

- `(WRP, 23) The child process` *<cmd>* `with pid` *<pid>* `could not be killed due to errno` *<errno>*`, reason:` ***reason***.

  The child process with pid *pid* could not be killed due to reason: *reason*.

  Action:
  Take action based on the reason *reason*.

- `(WRP, 24) Unknown flag option set for 'killChild'.`

  The killChild routine accepts one of the 2 flags: KILL_CHILD and DONTKILL_CHILD. If an option other than these two has been specified, this message is the result.

  Action:
  Please contact professional services.

- `(WRP, 25) Child process` *<cmd>* `with pid` *<pid>* `has exceeded its timeout period. Will attempt to kill the child process.`

  The child process *cmd* has exceeded its timeout period.

  Action:
  Please contact professional services.

- `(WRP, 28) RMS monitor has encountered an irregular sequence of timer interrupts, off by` *<offset>* `seconds. This may have been caused by a manual OS time change, or by an unusually high OS performance load, or by some other OS condition. If this error appears frequently, then normal RMS operations can no longer be guaranteed; it can also lead to a loss of heartbeats with remote hosts and to an elimination of the current host from the cluster.`

The RMS base monitor keeps track of the regularity of its timer interrupts that are supposed to occur every second. If the interrupts become irregular due to a high load, manual time change, or any other reason, the above notice is printed. If the discrepancy value becomes too high, or if this error appears frequently, then this might lead to a malfunction of the RMS base monitor, which can cause a loss of High Availability.

Action:

Do not attempt to change the system date/time by any significant value while RMS is running. Raise the priority of the RMS base monitor to ensure that it has enough CPU time to perform its operations during a high load.

- `(WRP, 29) RMS on the local host has received a message from host` ***host***`, but the local host is unable to resolve the sending host`s address. This could be due to a misconfiguration. This message will be dropped. Further such messages will appear in the switchlog.`

  RMS on the local host has received a message from host *host* whose address is not resolvable by the local host.

  Action:

  Make sure that the local host is able to resolve the remote host *host*'s address by checking for any misconfigurations.

- `(WRP, 30) RMS on the local host has received a message from host` ***host***`, but the local host is unable to resolve the sending host's address. This message will be dropped. Please check for any misconfiguration.`

  RMS on the local host has received a message from host *host* whose address is not resolvable by the local host.

  Action:

  Make sure that the local host is able to resolve the remote host *host*'s address by checking for any misconfigurations.

- `(WRP, 31) RMS has received a message from host` ***host*** `with IP address` ***receivedip***`. The local host has calculated the IP address of that host to be` ***calcip***`. This may be due to a misconfiguration in /etc/hosts. Further such messages will appear in the switchlog.`

The local host has received a message from host *host* with IP address *receivedip*, which is different from the locally calculated IP address for that host.

Action:
Check `/etc/hosts` for any misconfiguration.

- `(WRP, 32) RMS has received a message from host` ***host*** `with IP address` ***receivedip***`. The local host has calculated the IP address of that host to be` ***calcip***`. This may be due to a misconfiguration in /etc/hosts.`

  The local host has received a message from host *host* with IP address *receivedip*, which is different from the locally calculated IP address for that host. This message will be printed in the switchlog for every 25 such messages that have been received as long as the number of received messages is less than 500, if not this message is printed for every 250th such message received.

  Action:
  Check `/etc/hosts` for any misconfiguration.

- `(WRP, 33) Error while creating a message queue with the key` *<id>*`, errno =` *<errno>*`, explanation:` *<explanation>*`.`

  An abnormal OS condition occurred while creating a message queue.

  Action:
  Check OS conditions that affect memory allocation for message queues, such as the size of swap space, the values of parameters `msgmax`, `msgmnb`, `msgmni`, `msgtql`. Check if the maximum number of message queues have already been allocated.

- `(WRP, 34) Cluster host` ***host*** `is no longer in time sync with local node. Sane operation of RMS can no longer be guaranteed. Further out-of-sync messages will appear in the syslog.`

  The time on *host* is not in sync with the time on the local node.

  Action:
  Sync the time on *host* with the time on the local node.

- `(WRP, 35) Cluster host` ***host*** `is no longer in time sync with local node. Sane operation of RMS can no longer be guaranteed.`

  The time on the cluster host *host* differs significantly (> 25 seconds) from the local node.

  Action:
  Make sure that all the cluster hosts are in time sync.

- `(WRP, 42) The interconnect <`*interconnect*`> to cluster host <`*host*`> has failed.`

  The interconnect *interconnect* has failed to host *host* has failed.

  Action:
  Fix the interconnect.

9 Fatal error messages

This chapter contains a detailed list of all fatal RMS error messages that appear in the switchlog. Most messages are accompanied by a description of the probable cause(s) and a suggested action to correct the problem. In some cases, the description or action is self-evident and no further information is necessary.

Some messages in the listings that follow contain words printed in *italics*. These words are placeholders for values, names, or strings that will be inserted in the actual message when the error occurs.

RMS error code description

A prefix in each message contains an error code and message number identifying the RMS component that detected the problem. You may need to provide this prefix to support engineers who are diagnosing your problem. The following list summarizes the possible error codes and the associated component:

`ADC`:	Admin configuration
`ADM`:	Admin, command, and detector queues
`BM`:	Base monitor
`CML`:	Command line
`CMM`:	Communication
`CRT`:	Contracts and contract jobs
`DET`:	Detectors
`INI`:	init script
`MIS`:	Miscellaneous
`QUE`:	Message queues
`SCR`:	Scripts
`SYS`:	SysNode objects
`UAP`:	userApplication objects
`US`:	us files
`WLT`:	Wait list
`WRP`:	Wrappers

9.1 ADC: Admin configuration

- `(ADC, 16) Because some of the global environment variables were not set in hvenv file - RMS cannot start up. Shutting down.`

  All of the global environment variables `RELIANT_LOG_LIFE`, `RELIANT_SHUT_MIN_WAIT`, `HV_CHECKSUM_INTERVAL`, `HV_LOG_ACTION_THRESHOLD`, `HV_LOG_WARNING_THRESHOLD`, `HV_WAIT_CONFIG` and `HV_RCSTART` have to be set in the `hvenv` in order for RMS to function properly. If some of them have not been set, RMS exits with exit code 1.

  Action:
  Set the values of all the environment variables in `hvenv`.

- `(ADC, 21) Because some of the local environment variables were not set in hvenv file, RMS cannot start up. Shutting down.`

  If some of the local environment variables have not been set in the `hvenv` file, RMS prints this message and exits with exit code 1.

  Action:
  Make sure that all the local environment variables have been set to an appropriate value in the `hvenv` file.

- `(ADC, 69) RMS will not start up - previous errors opening file.`

  The previous error was a failure to open the file needed for dynamic startup. The base monitor will exit.

  Action:
  Verify the file existence and reissue dynamic startup request.

9.2 ADM: Admin, command, and detector queues

- `(ADM, 1) cannot open admin queue.`

RMS uses UNIX message queues for interprocess communication. The admin queue is one such queue used for communication between utilities like `hvutil`, `hvswitch`, etc. If there is a problem opening this queue, then this message is printed and RMS exits with exit code 3.

Action:
Contact field support.

- `(ADM, 2) RMS will not start up - errors in configuration file.`

  When RMS is starting up, it performs dynamic modification under the hood, if during this phase it encounters errors in its configuration file, RMS exits with exit code 23.

  Action:
  Make sure there are no errors in the configuration file based on the error messages printed prior to the above message in the switchlog.

9.3 BM: Base monitor

- `(BM, 3) Usage:` ***progname*** `[-c config_file] [-m] [-h time] [-l level] [-r count] [-w time] [-n]`

  If RMS has not been invoked in the right way because either some arguments were missing or haven't been used correctly, this message is printed out to the switchlog indicating the arguments. RMS exits with exit code 3.

  Action:
  Start RMS with the right arguments.

- `(BM, 49) Failure calculating configuration checksum.`

  During dynamic reconfiguration, RMS calculates the configuration checksum by using `/usr/bin/sum`. If this fails, then this message is printed and RMS exits with the exit code 52.

  Action:
  Check if `/usr/bin/sum` is available.

- `(BM, 51) The RMS-CF interface is inconsistent and will require operator intervention. The routine "`*routine*`" failed with errno `*errno*` - "`*errorreason*`"`

  While setting up CF, if RMS encounters a problem in the routine *routine* that can either be "dlopen" or "dlsym", it exits with exit code 95 or 94 respectively. The *errorreason* gives the reason for the error.

  Action:
  Contact field support.

- `(BM, 58) Not enough memory -- RMS cannot continue its operations and is shutting down.`

  This is a generic message that is printed out to the switchlog before RMS discontinues its functioning because it does not have enough memory for it to operate.

  Action:
  Contact field support.

- `(BM, 67) An error occurred while writing out the RMS configuration after dynamic modification. RMS is shutting down.`

  Upon concluding dynamic modification, RMS dumps out its current configuration into a file `/var/tmp/config.us`. If this cannot be done, RMS cannot recalculate configuration's checksum. Therefore, it shuts down.

  Action:
  The previous message in switchlog explains why RMS has not been able to write down the configuration file. Please correct the host environment according to the description, or contact field support."

- `(BM, 69) Some of the OS message queue parameters msgmax=`*<msgmax>*`, msgmnb=`*<msgmnb>*`, msgmni=`*<msgmni>*`, msgtql=`*<msgtql>* `are below lower bounds `*<hvmsgmax>*`, `*<hvmsgmnb>*`, `*<hvmsgmni>*`, `*<hvmsgtql>*`. RMS is shutting down.`

  One or more of the system defined message queue parameters is not sufficient for correct operation of RMS. RMS shuts down with exit code 28.

  Action:

Change the OS message queue parameters and reboot the OS before restarting RMS.

- `(BM, 82) A message to host` <*remotehost*> `failed to reach that host after` <*count*> `delivery attempts. Communication with that host has been broken. Therefore, RMS monitor on this host` <*localhost*> `is going down.`

  A communication breakdown prevented delivery of a message between the local and remote RMS monitors. In this case the local monitor exits.

  Action:
  Make sure *remotehost* is up and that communication between the two hosts is possible. Use standard tools such as `ping` and make sure that the local root account can `rlogin` or `rsh` to the remote host. After communication has been re-established, restart the local RMS monitor.

- `(BM, 89) The SysNode length is` ***length***`. This is greater than the maximum allowable length of` ***maxlength***`. RMS will now shut down.`

  The `SysNode` name length is greater than the maximum allowable length.

  Action:
  Ensure that the length of the `SysNode` name is less than *maxlength*.

9.4 CML: Command line

- `(CML, 14) ###ERROR Unable to find or Invalid configuration file.### #####CONFIGURATION MONITOR exits !!!!!######`

  The configuration file specified for RMS is non-existent. RMS exits with exit code 1.

  Action:
  Specify a valid configuration file for RMS to function.

9.5 CMM: Communication

- `(CMM, 1) Error establishing outbound network communication.`

If there is an error in creating outbound network communication, this message is the result and RMS exits with exit code 12.

Action:
System error. Contact field support.

- `(CMM, 2) Error establishing inbound network communication.`

  If there is an error in creating inbound network communication, this message is the result and RMS exits with exit code 12.

  Action:
  System error. Contact field support.

- `(CMM, 3) Create queue error NODE_SYS_Q.`

  The `NODE_SYS_Q` is used by the RMS base monitor to communicate the list of `SysNode` objects to `hvdet_node`. If there is a problem creating this queue for some reason, RMS exits with exit code 12.

  Action:
  Contact field support.

9.6 CRT: Contracts and contract jobs

- `(CRT, 6) Fatal system error in RMS. RMS will shut down now. Please check the bmlog for SysNode information.`

  A system error has occurred within RMS.

  Action:
  Please contact field support.

9.7 DET: Detectors

- `(DET, 8) Failed to create DET_REP_Q.`

  If RMS is unable to create the Unix Message queue DET_REP_Q for communication between a detector and itself, this message is the result and RMS exits with exit code 12.

  Action:
  Contact field support.

- `(DET, 9) Message send failed in detector request Q:` *queue*`.`

  During `hvlogclean`, the detector request queue *queue* is used for sending information to the detector from the base monitor. If there is a problem in communication, this message is the result and RMS exits with exit code 12.

  Action:
  Contact field support.

- `(DET, 16) Cannot create gdet queue of kind g`*kind*`.`

  Each of the generic detectors has a message queue, which it uses to communicate with the base monitor. If there is a problem creating a queue for a detector of kind *kind*, this message is the result and RMS exits with exit code 12.

  Action:
  Contact field support.

- `(DET, 18) Error reading hvgdstartup file. Error message:` *errorreason*`.`

  When the RMS base monitor tries starting up the generic detectors, it parses the `hvgdstartup` file for detector information. If RMS encounters an error while reading this file, it prints this message along with the reason *errorreason* for the failure. RMS then exits with exit code 26.

  Action:
  Contact field support.

9.8 INI: init script

- `(INI, 4) InitScript does not have execute permission.`

  InitScript exists, but cannot be executed.

  Action:
  make InitScript executable.

- `(INI, 7)` *sysnode* `must be in your configuration file.`

  If the local `SysNode` *sysnode* is not part of the configuration file, this message is the result and RMS exits with exit code 23.

Action:
Make sure that the local `SysNode` *sysnode* is part of the configuration file.

- `(INI, 10) InitScript has not completed within the allocated time period of` *timeout* `seconds.`

  InitScript was still running when the time period allocated for its execution has expired. The timeout period is the least of the values defined in the environment variable `SCRIPTS_TIME_OUT` in the `hvenv` file, or 300.

  Action:
  Increase the timeout value, or correct the conditions lead to timeout during script execution.

- `(INI, 11) InitScript failed to start up - errno` *errno*`, reason:` *reason*`.`

  An error occurred during startup of InitScript. The errno code <*errno*> and reason <*reason*> are presented in the message.

  Action:
  Correct the erroneous host condition for InitScript to be able to start up.

- `(INI, 12) InitScript returned non-zero exit code` *exitcode*`.`

  InitScript completed with a non-zero exit code <*exitcode*>.

  Action:
  Correct the erroneous host condition for InitScript to be able to return a zero exit code, or fix the InitScript itself.

- `(INI, 13) InitScript has been stopped.`

  InitScript has been stopped.

  Action:
  Correct the erroneous host condition for InitScript to run without stopping, or fix the InitScript itself.

- `(INI, 14) InitScript has been abnormally terminated.`

  InitScript has been abnormally terminated.

  Action:
  Correct the erroneous host condition for InitScript to run without stopping, or fix the InitScript itself.

9.9 MIS: Miscellaneous

- `(MIS, 4) The locks directory` *directory* `cannot be cleaned of all old locks files: at` *call*`, errno =` *errnonumber*`, error --` *errortext*`.`

  The various RMS commands like `hvdisp`, `hvswitch`, `hvutil` and `hvdump` utilize the lock files from the directory *directory* for signal handling purposes. These files are deleted after these commands are completed. The locks directory is also cleaned when RMS starts up. If they are not cleaned for some reason, this message is the result. RMS exits with exit code 99. The *call* indicates at which stage the cleanup has failed, *errornumber* is the OS errno value, *errortext* is the OS supplied explanation for the errno.

  Action:
  Make sure that the locks directory *directory* exists.

9.10 QUE: Message queues

- `(QUE, 1) Error status in ADMIN_Q.`

  Different utilities use the ADMIN_Q to communicate with the base monitor. If there is an error with this queue, this message is the result and RMS exits with exit code 3.

  Action:
  Contact field support.

- `(QUE, 2) Read message failed in ADMIN_Q.`

  This message is the result of the RMS base monitor being unable to extract a message of the ADMIN_Q that is used for communication between the utilities and RMS. RMS then exits with exit code 3.

  Action:
  Contact field support.

- `(QUE, 5) Network message read failed.`

  If there is a problem reading a message over the network, this error is the result and RMS exits with exit code 3.

  Action:

System error. Contact field support.

- `(QUE, 6) Network problem occurred.`

  This message is the result of a network problem occurring when transferring messages.

  Action:
  System error. Contact field support.

- `(QUE, 11) Read message failed in DET_REP_Q.`

  All the detectors use the queue DET_REP_Q to communicate with the RMS base monitor. If there is a problem in reading the message of the queue, RMS prints this message and exits with exit code 15.

  Action:
  Contact field support.

- `(QUE, 12) Error status in DET_REP_Q:` *status*`.`

  This message is the result of the RMS base monitor having a problem with the queue DET_REP_Q that is used by the different detectors to report their state. RMS then exits with exit code 15.

  Action:
  Contact field support.

9.11 SCR: Scripts

- `(SCR, 4) Failed to create a detector request queue for detector` *detectorname*`.`

  If a detector request queue could not be created for detector *detector_name*, this message is the result and RMS exits with exit code 12.

  Action:
  System problem. Contact field support.

- `(SCR, 5) REQUIRED PROCESS RESTART FAILED: Unable to restart` *detector*`. Shutting down RMS.`

If the detector *detector* could not be restarted, this message is the result with RMS shutting down with exit code 14. The restart could have failed for any of the following reasons:

– If the detector needs to be restarted more than 3 times in one minute.
– If there is a problem with memory allocation within RMS.

Action:
Contact field support.

- `(SCR, 10) InitScript did not run ok. RMS is being shut down.`

RMS runs the InitScript initially. The value of InitScript is the value of the environment variable `RELIANT_INITSCRIPT` in `hvenv`. For some reason, if this InitScript fails (like exiting with a non-zero code, getting a signal, etc.), then this message is printed and RMS shuts down with exit code 56.

Action:
Contact field support.

- `(SCR, 12) incorrect initialization of RealDetReport; Shutting down RMS.`

Since the scripts are executed based on the reports of the detectors, if the detector reports a state other than Online, Offline, Faulted, Standby or NoReport, this message is the result with RMS exiting with exit code 8.

Action:
Make sure that the detector only reports states Online, Offline, Faulted, Standby or NoReport.

- `(SCR, 13) ExecScript: Failed to exec script` <*script*> `for object` <*nodename*>`: errno` *errno*`.`

RMS has been unable to execute a script <*script*> for the object <*objectname*>. The error number *errno* returned by the operating system provides a diagnosis of the failure. RMS exits with exit code 8.

Action:
Consult the system manual pages or the appendix of this manual for the explanation for error number *errno* and see if the cause is evident. If not, contact field support.

- `(SCR, 15) node_sys_q cannot be accessed.`

The queue node_sys_q is used by the detector `hvdet_node` to get the list of the `SysNode` objects from the RMS base monitor, if there is some problem with this queue, this message is printed and RMS exits with exit code 12.

Action:
Contact field support.

- `(SCR, 18) Message send failed to node_sys_q.`

  The RMS base monitor uses the queue `node_sys_q` to send the list of `SysNode` objects to `hvdet_node` after `hvmod` (the initial one on startup or the subsequent ones when hvmod has been invoked explicitly). If RMS is unable to send this information to hvdet_node, this message is printed and RMS exits with exit code 2.

  Action:
  Contact field support.

- `(SCR, 26) The sdtool notification script has failed with status` *status* `after dynamic modification.`

  After dynamic modofication Shutdown Facility is notified via sdtool about the changes in the current configuration. If sdtool exits abnormally, then the base monitor must exit.

  Action:
  Verify that sdtool and Shutdown Facility are operating properly.

9.12 SYS: SysNode objects

- `(SYS, 33) The RMS cluster host` <*hostname*> `does not have a valid entry in the /etc/hosts file. The lookup function gethostbyname failed. Please change the name of the host to a valid /etc/hosts entry and then restart RMS.`

  If the lookup function gethostbyname which searches the file `/etc/hosts` to get information about the host *hostname* is unable to find a valid entry for it, this message is printed and RMS exits with exit code 114.

  Action:

Make sure that the host name *hostname* has a valid entry in `/etc/hosts` and restart RMS.

- `(SYS, 52) SysNode` *sysnode*`: error creating necessary message queue NODE_REQ_Q...exiting.`

  When RMS encounters a problem in creating the NODE_REQ_Q, this message is the result and RMS exits with exit code 12.

  Action:
  Contact field support.

9.13 UAP: userApplication objects

- `(UAP, 36)` *object*`: double fault occurred, but Halt attribute is set. RMS will exit immediately in order to allow a failover!`

  When the Halt attribute is set for an object and a double fault occurs, then RMS will exit with code 96 on that node.

  Action:
  Contact field support.

9.14 US: us files

- `(US, 1) RMS will not start up - fatal errors in configuration file.`

  Errors were found in the configuration file that prevented RMS startup. This is usually caused by manual editing or distribution of the configuration file.

  Action:
  Use only PCS or the Wizard Tools to create and activate your configuration. If you have used only the standard tools and this error persists, contact field support.

- `(US, 42) A State transition error occured. See the next message for details.`

A state transition error occured in the course of RMS state transitons. Details of the error are printed in the subsequent lines.

Action:
Save the error description and contact field support.

9.15 WLT: Wait list

- `(WLT, 9) sdtool notification timed out after` <*timeout*> `seconds.`

  After dynamic modofication, the Shutdown Facility is notified via sdtool about the changes in the current configuration. If this notification does not finish within the period specified by the local `SysNode` `ScriptTimeout` value, the base monitor must exit.

  Action:
  Verify that sdtool and Shutdown Facility are properly operating. Increase the `ScriptTimeout` value if needed.

9.16 WRP: Wrappers

- `(WRP, 40) The length of the` ***type*** `name specified for the host` ***host*** `is` <*length*> `which is greater than the maximum allowable length` <*maxlength*>`. RMS will exit now.`

  The length of the interconnect name is greater than the maximum value.

  Action:
  Make sure that the interconnect name is less than the maximum value of *maxlength*.

10 Console error messages

This chapter contains a detailed list of all RMS error messages that appear on the console. The messages are listed here in alphabetical order; messages that begin with replaceable strings are listed first. Most messages are accompanied by a description of the probable cause(s) and a suggested action to correct the problem. In some cases, the description or action is self-evident and no further information is necessary.

Some messages in the listings that follow contain words printed in *italics*. These words are placeholders for values, names, or strings that will be inserted in the actual message when the error occurs.

10.1 Console messages in alphabetical order

- *command1* `cannot get list of resources via` <*command2*> `from hvcm.`

  The wizards rely on `hvmod` for dynamic modification. If there is a problem executing command *command2*, this message is the result and `hvmod` exits with exit code 15.

  Action:
  Contact field support.

- *command* `failed due to errors in` <*argument*>`.`

  When `hvmod` has been invoked, it uses `hvbuild` internally, if there is a problem with the execution of `hvbuild`, this message is the result and `hvmod` is aborted. `hvmod` then exits with exit code 1.

  Action:
  Contact field support.

- *command*`: bad state:` *state*`.`

  If `hvassert` is performed for a state *state* which is not among the states that can be asserted, this message is the result and `hvassert` exits with exit code 1.

  Action:
  Make sure that the state specified for `hvassert` is assertable.

- *command*: `bad timeout:` *timeout*.

  If the timeout specified for the `hvassert` command is not a number, this message is the result and the utility exits with exit code 1.

  Action:
  Specify a number for the timeout value of `hvassert`.

- *command*: `cannot open file` *filename*.

  `hvsend` is used to send messages to an object in a resource graph. It can get the list of messages to send from a file. If this file cannot be opened, this message is the result and the `hvsend` utility exits with exit code 8.

  Action:
  Make sure that the file *filename* exists.

- *command*: `could not create a pipe`

  If the utility *command* could not open the tty to be written to, this message is the result and the utility exits with exit code 7.

  Action:
  Contact field support.

- *command*: `failed due to undefined variable: local_host.`

  If the `hvsend` utility is unable to find the value of the environment variable `RELIANT_HOSTNAME`, this message is the result and it exits with exit code 7.

  Action:
  Make sure that `RELIANT_HOSTNAME` is defined.

- *command*: `file already exists`

  When '`hvdisp -o`' has been invoked by the user and the output file that has been specified as an argument already exists, this message is the result and `hvdisp` exits with exit code 6.

  Action:
  Specify a filename that does not already exist as the argument to '`hvdisp -o`'.

- *command*: `message queue is not ready yet!`

  The command *command* relies on a message queue to transmit messages to the RMS base monitor. If this message queue is not available for some reason, this message is the result and the utility exits with exit code 3.

  Action:
  Contact field support.

- *command*: `Must be super-user to issue this command`

  This message indicates that in order to run the command *command*, the user should have root privileges.

  Action:
  Make sure that the user has root privileges before issuing the command.

- *command*: `RMS is not running`

  When the command *command* has been invoked, it checks to make sure that RMS is running, if not this message is the result and the utility exits with exit code 2.

  Action:
  Make sure that RMS is running before invoking the different utilities.

- *directory*: `cannot put message in queue`

  The various RMS commands like `hvdisp`, `hvswitch`, `hvutil` and `hvdump` utilize the lock files from the directory *directory* for signal handling purposes. These files are deleted after these commands are completed. The locks directory is also cleaned when RMS starts up. If they are not cleaned for some reason, this message is the result. RMS exits with exit code 99.

  Action:
  Make sure that the locks directory *directory* exists.

- *resource* `is not in state` *state*`.`

  If the `hvassert` on an object *resource* for a state *state* discovers that the resource is not in that state, this message is printed and `hvassert` exits with exit code 1.

  Action:
  None required.

- *timestamp*`: NOTICE: User has been warned of 'hvshut -f' and has elected to proceed.`

  When the user invokes '`hvshut -f`', and then has elected to proceed with the command then, this message is printed to confirm that '`hvshut -f`' is being invoked.

  Action:
  None required.

- <*command*> `failed with exit code` *exitcode*

  When the `hvlogclean` utility is invoked without the `-d` option, it executes the command *command*, if this command could not be executed for some reason, it returns the exit code *exitcode* and then the utility exits with exit code 6.

  Action:
  Take action based on the exit code *exitcode*.

- `Assertion condition failed.`

  If `hvassert` fails while using `-f` or `-F` options, this message is printed and `hvassert` exits with exit code 1.

  Action:
  None required.

-
  ```
  BEWARE: 'hvshut -f' may break the consistency of the
  cluster.
  No further action may be executed by RMS until the
  cluster consistency is re-established. This re-estab-
  lishment
  includes restart of RMS on the shut down host.
  Do you wish to proceed? (yes = shut down RMS / no = leave
  RMS running).
  ```

This is a message asking for confirmation from the user if he wants to proceed with '`hvshut -f`'. If the user elects to proceed, `yes` would be the appropriate answer, and a `no` if there is no intention of going ahead with it.

Action:
Respond to the prompt.

- `BEWARE: the hvreset command will result in a reinitialization of the graph of the specified userApplication. This affects basically the RMS state engine only. The re-initialization does not mean, that activities invoked by RMS so far will be made undone. Manual cleanup of halfway configured resources may be necessary.`
  `Do you wish to proceed? (yes = reset application graph / no = abort hvreset).`

Action:
Respond to the prompt.

- `Can't open modification file.`

When `hvmod` is invoked with the `-c` option, it utilizes a temporary file, if this file cannot be opened for writing, this message is the result and `hvmod` exits with exit code 1.

Action:
Contact field support.

- `Cannot start RMS! BM is currently running.`

RMS is already running on the local host.

Action:
Shut down the currently running version of RMS and restart.

- `Change dest_object to` *node*`.`

Action:
(None specified)

- `Command aborted.`

  A command has prompted for reconfirmation. If the user answers with a `no` to the question of whether he wants to proceed with the command, this message is printed and the command is aborted.

  Action:
  None required.

- `Command timed out!`

  Action:
  (none specified)

- `Could not open localfile or could not create temporary file` *filename*

  If during `hvrcp`, the localfile cannot be opened for reading or the temporary file *filename* cannot be opened for writing, this message is printed and `hvrcp` exits with exit code 7.

  Action:
  Check the permissions on the localfile to make sure that it is readable.

- `Could not restart RMS.  RELIANT_PATH not set.`

  When the detector restarts RMS, it checks the value of the environment variable `RELIANT_PATH`, if it cannot get the value of this variable, this message is printed.

  Action:
  Make sure that `RELIANT_PATH` is set to an appropriate value.

- `Delay` *delay* `seconds.....`

  This is an informational message specifying the delay *delay* in seconds that `hvsend` has been provided.

  Action:
  None required.

- `DISCLAIMER: The hvdump utility will collect the scripts, configuration files, log files and any core dumps. These will be shipped out to RMS support. If there are any proprietary files you do not want included, please exit now. Do you want to proceed? (yes = continue / no = quit)`

  This message is printed out on executing '`hvdump -E`' and will collect the necessary information only if the answer to the above question is "yes".

  Action:
  Respond to the prompt.

- `DISCLAIMER: The hvdump utility will now collect the necessary information. These will be shipped to RMS support.`

  This message just indicates that the `hvdump` utility will now start collecting the information.

  Action:
  None required.

- `Dynamic modification is in progress, can't assert states.`

  It is not possible to perform an `hvassert` when dynamic modification is in progress.

  Action:
  Perform `hvassert` after dynamic modification finishes.

- `Error becoming a real time process:` *errorreason*

  The RMS base monitor runs as a real time process (thereby giving it higher priority over other processes) on Solaris. If there is a problem in the base monitor becoming a real time process due to *errorreason*, then this message is the result.

  Action:
  Take action based on the reason.

- `Error setting up real time parameters:` *errorreason*

  If there is a problem while setting up the parameters for the RMS base monitor to run as a real-time process, this message is the result along with the reason *errorreason* for the problem.

  Action:
  Take action based on the reason.

- `Error while starting up bm on the remote host` <*targethost*>`:` *errorreason*

  When `hvcm` is invoked with the `-s` option to start RMS on a remote host <*targethost*>, if there is a problem in starting up RMS on the remote host, this message is the result along with the reason for the problem <*error-reason*>.

  Action:
  Take action based on the reason for the problem and reissue '`hvcm -s`'.

- `Error while starting up local bm:` *errorreason*

  Error while starting up local bm: *errorreason*

  Action:
  Take action based on the reason.

- `Failed to dup a file descriptor.`

  If RMS is unable to dup a file descriptor while setting the environment, this message appears.

  Action:
  Contact field support.

- `Failed to exec the hvenv file` <*hvenvfile*>`.`

  RMS was unable to `exec` the `hvenv` environment variable file *hvenvfile*.

  Action:
  Contact field support.

- `Failed to open pipe.`

  If RMS is unable to open a pipe for communication, this message is the result and RMS exits with exit code 1.

  Action:
  Contact field support.

- `FATAL ERROR Could not restart RMS.  Restart count exceeded.`

  When the detector tries to restart RMS, it keeps track of how many times RMS had to be restarted. If this count has exceeded 3, then this message is the result.

  Action:
  Contact field support.

- `FATAL ERROR: Could not restart RMS.  Restart script (`*script*`) does not exist.`

  When the detector is unable to restart RMS because the script *script* is non-existent, this message is the result.

  Action:
  Make sure that the script *script* exists.

- `FATAL ERROR: Could not restart RMS. Failed to recreate RMS restart count file.`

  When the detector tries to restart RMS, it keeps track of how many times RMS had to be restarted by writing out the necessary information out to a count file. If this file cannot be opened for writing the above message is printed.

  Action:
  Contact field support.

- `FATAL ERROR: RMS has failed to start!`

  Internal error.

  Action:
  Contact field support.

- `File open failed (`*path*`):` *errorreason*`.`

  If the file *path* that is used by the `hvassert` utility to communicate with the RMS base monitor could not be opened, this message is the result, along with the reason *errorreason* for this failure. `hvassert` then exits with exit code 5.

  Action:
  Contact field support.

- `Forced shut down on the local cluster host!`

  When the detector restarts the base monitor, it prints this message before proceeding.

  Action:
  None required.

- `Fork failed.`

  If RMS is unable to fork a process, it prints this message and exits with exit code 1.

  Action:
  Contact field support.

- `hvsend: dest_object is not specified.`

  If `hvsend` has been provided an unknown option in the input file, this message is printed and `hvsend` exits with exit code 9.

  Action:
  Make sure that you specify a valid option.

- `hvutil: Could not determine if RMS is running on <`*targethost*`>, errno` *exitcode*

  Printed when '`hvutil -A` *targethost*' is called indicating that the command failed to ascertain whether or not RMS is running on *targethost*. The *exitcode* indicates a value in /usr/include/sys/errno.h.

  Action:
  Depends on the *exitcode* value

- `hvutil: Could not determine IP address of` <*targethost*>

  The name of the cluster host could not be resolved to an IP address.

  Action:
  Add an entry for *targethost* into the `/etc/hosts` file of all cluster hosts.

- `hvutil: debug option must be a positive number for on, 0 for off.`

  When '`hvutil -L`' has been invoked with a loglevel that is not one of `0` or `1`, this message is the result and it exits with exit code 6.

  Action:
  Specify a valid logging level of 0 or 1 for the utility.

- `hvutil: Detector time period must be greater than` *minimumtime* `.`

  If the detector time period specified as an argument with '`hvutil -t`' is less than *minimumtime*, `hvutil` is aborted and exits with exit code 5.

  Action:
  Invoke `hvutil` with a time period that is greater than *minimumtime*.

- `hvutil: Failed to allocate socket`

  Failed to allocate a socket to communicate with a remote host.

  Action:
  Contact professional services to determine the cause.

- `hvutil: Missing /etc/services entry for "rmshb"`

  An entry is missing in the `/etc/services` file for the RMS heartbeat.

  Action:
  Add an entry on all cluster hosts for rmshb using tcp

- `hvutil: Notify string is longer than` *mesglen* `bytes`

  Notify string is too long.

  Action:
  Notify string should not be longer than *mesglen* bytes.

- `hvutil: RMS is not running on <`*targethost*`>`

  Printed when '`hvutil -A` *targethost*' is called indicating that RMS is not running on the named host.

  Action:
  None required.

- `hvutil: RMS is running on <`*targethost*`>`

  Printed when '`hvutil -A` *targethost*' is called indicating that RMS is running on the named host.

  Action:
  None required.

- `hvutil: The resource <`*resource*`> does not have a detector associated with it`

  The resource *resource* does not have a detector.

  Action:
  Issue '`hvutil -N`' on a resource which has a detector.

- `hvutil: The resource <`*resource*`> is not a valid resource`

  The resource *resource* is not a valid resource.

  Action:
  Issue '`hvutil -N`' on a resource which has a detector and is part of the resource graph.

- `hvutil: time period of detector must be an integer.`

  If the detector time period specified as an argument with '`hvutil -t`' is not a number, `hvutil` is aborted and exits with exit code 6.

  Action:
  Make sure that the detector time period is an integer.

- `hvutil: Unable to open the notification file` <*path*> `due to reason:` *reason*

  `hvutil` was unable to open the file *path* because of *reason*.

  Action:
  Contact field support.

- `Invalid delay.`

  If the delay specified for sending a message using `hvsend` is a number less than zero, this message is printed.

  Action:
  Provide a valid value for the delay.

- `It may take few seconds to do Debug Information collection.`

  As the `hvdump` utility dumps out the information regarding the resource graph, it prints this message while it is collecting the information.

  Action:
  None required.

- `localfile` *filename* `does not exist or is not an ordinary file`

  If the localfile specified as an argument to `hvrcp` does not exist or if it is not a regular file, `hvrcp` exits with exit code 7.

  Action:
  Make sure that the localfile exists and is an ordinary file.

- `Modification file name is missing on the command line, usage: hvmod [-i] [-l] -f config_file.us | -E | -L | [-i] [-l] -c "modification directives"`

  When the `hvmod` utility is invoked with an option that does not conform to its expected usage this message is the result and the utility exits with exit code 2.

  Action:
  Follow the expected usage for the utility.

- `Name of the modification file is too long.`

  If the name of the modification file specified as an argument through the `-f` option or the modification directives specified via the `-c` option are greater than 113, this message is printed and `hvmod` exits with exit code 4.

  Action:
  Make sure that the arguments specified via `-f` and `-c` options are not too long.

- `NOTICE: User has been warned of 'hvshut -f -a' and has elected to proceed.`

  When the user has elected to proceed with the command '`hvshut -f -a`' this message is printed to confirm the choice.

  Action:
  None required.

- `NOTICE: User has been warned of 'hvshut -L' and has elected to proceed.`

  When the user invokes '`hvshut -L`', and then has elected to proceed with the command then, this message is printed to confirm that '`hvshut -L`' is being invoked.

  Action:
  None required.

- `RELIANT_LOG_PATH is not defined`

  When the `hvlogclean` utility is invoked without the `-d` option, it needs the value of the environment variable RELIANT_LOG_PATH, to get to the `hvloginit` script. If the value of the variable cannot be found, this message is the result and the utility exits with exit code 6.

  Action:
  Make sure that the environment variable RELIANT_LOG_PATH has not been unset and is set to the appropriate value.

- `RELIANT_PATH is not defined`

  When the `hvlogclean` utility is invoked without the `-d` option, it needs the value of the environment variable `RELIANT_PATH`, to get to the `hvloginit` script. If the value of the variable cannot be found, this message is the result and the utility exits with exit code 6.

  Action:
  Make sure that the environment variable `RELIANT_PATH` is set to the appropriate value.

- `Remote host` <*hostname*> `is not Online.`

  When performing `hvassert`, if the remote host *hostname* is not Online, this message is printed and `hvassert` exits with exit code 1.

  Action:
  Make sure that the remote host is Online before performing `hvassert`.

- `Remote host does not exist -` *host*`.`

  If the `SysNode` specified as '`hvassert -h` *host* `...`' is not part of the RMS resource graph, this message is printed and `hvassert` exits with exit code 10.

  Action:
  Make sure that the remote hostname specified for `hvassert` exists.

- `Remote system is not online.`

  Trying to perform `hvassert` on an object on a remote host which does not have RMS running, causes this message and `hvassert` exits with exit code 10.

  Action:
  Make sure that the remote system has RMS running before performing `hvassert`.

- `Reset of RMS has been aborted.`

  When the user invokes `hvreset`, the `hvreset` utility asks for a confirmation. If the answer is not `yes` then `hvreset` is aborted and this message is printed out.

  Action:
  None required.

- `Resource does not exist -` *resource*`.`

  If there is an attempt to perform `hvassert` on a resource which is not part of the RMS resource graph, this message is printed and `hvassert` then exits with exit code 10.

  Action:

  Make sure that a resource exists before performing `hvassert` on it.

- `RMS has failed to start!`
  `'hvcm' has been invoked without specifying a configuration with the -c attribute, but with specifying other command line options. This may cause ambiguity and is therefore not possible. Please specify the entire commandline or use 'hvcm' without further options to run the default configu-ration`

  This message appears when the user tries to start RMS without the `-c` option and specifying other commandline options.

  Action:

  When using `hvcm` with the `-c` option, '`-c` *configname*' should be the last arguments on the command line. Alternatively, to use with the default configuration, enter `hvcm` without any arguments to start RMS on the local node, and '`hvcm -a`' to start RMS aon all nodes.

- `RMS has failed to start!`
  `didn't find a valid entry in the RMS default configuration file "`*configfilename*`"`

  This message appears when the RMS default configuration file exists but does not contain a valid reference to a configuration to run.

  Action:

  Either place a default configuration file name in the RMS default config-uration file or put the current configuration name in it that the user wants to start.

- `RMS has failed to start!`
  `invalid entry in the RMS default configuration file "`*config-filename*`"`

  The user is not allowed to start RMS if the default configuration has invalid entry in the RMS default configuration file. The possible valid entries are 1. *configname* or 2. '`hvcm` *<options>* `-c` *<configname>*'. Refer to the `hvcm` man page for details on valid options in format 2.

  Action:
  Remove all invalid entries in the RMS default configuration file. Refer the `hvcm` man page.

- `RMS has failed to start!`
  `multiple entries in the RMS default configuration file "`*configfilename*`"`

  The user is not allowed to start RMS if there are multiple entries in the default configuration file config.us.

  Action:
  The user has to remove all the obscure entries in the RMS default configuration file and has to have only one valid configuration in it.

- `RMS has failed to start!`
  `RELIANT_HOSTNAME is not defined in the RMS environment`

  The environment variable `RELIANT_HOSTNAME` is not properly set.

  Action:
  Ensure that the RMS environment variable `RELIANT_HOSTNAME` wasn't set erroneously to "" (null string) or explicitly unset in `hvenv.local`.

- `RMS has failed to start!`
  `the number of arguments specified at the command line overrides the internal buffer of the RMS start utility`

  This message appears when the number of arguments specified at the command line is more than the buffer capacity (= 30 command line arguments).

  Action:
  Refer to the `hvcm` manual page for the correct syntax and usage.

- `RMS has failed to start!`
  `the number of arguments specified at the RMS default configuration file "`*configfilename*`" overrides the internal buffer of the RMS start utility`

  This message appears when the user tries to start the RMS using the RMS default configuration file but unable to do so because the number of arguments specified in the RMS default configuration file overrides the internal buffer of the RMS start utility.

  Action:
  Remove some of the unwanted arguments from the RMS default configuration file. Check the man page for `hvcm` to get the required options to start RMS.

- `RMS has failed to start!`
  `the options "-a" and "-s" are incompatible and may not be specified both`

  This message appears when the user tries to start RMS uses the options `-a` and `-s` simultaneously.

  Action:
  Check the man page for `hvcm` to get the format.

- `rms is dead`

  The `hvrcp` utility checks whether the RMS base monitor is alive every 10 seconds, if it finds that it is not alive, it prints this message and exits with exit code 1.

  Action:
  Get RMS running on the host.

- `RMS on node` ***node*** `could not be shutdown with hvshut -A.`

  Action:
  (none specified)

- `Root access required to start hvcm`

  To start RMS the user must have root access.

  Action:
  login as root and try `hvcm`.

- `Sending` *data* `to` *resource*`.`

  This message is printed when logging is turned on and *data* is being sent to the object *resource*.

  Action:
  None required.

- `Shutdown of RMS has been aborted.`

  When the user invokes '`hvshut -L`', the `hvshut` utility asks for a confirmation, if the answer is `no` then '`hvshut -L`' is aborted and this message is printed out.

  Action:
  None required.

- `Starting Reliant Monitor Services now`

  When RMS is starting up, this message is printed out.

  Action:
  None required.

- `Starting RMS on remote host` *host* `now`

  This message will be printed when RMS is being started on the remote host *host*.

  Action:
  None required.

- `startup aborted per user request`

  When RMS is being started up with the "-c" option, if the configuration file specified is different from the entry in CONFIG.rms, RMS asks for confirmation from the user, if he wants to activate the different configuration file. If the response is "no", then the aforementioned message is the result.

  Action:
  None required.

- `The command '`*command*`' could not be executed`

  The execution of the command *command* failed.

  Action:
  Check to see if the *command* is available.

- `The command '`*command*`' failed to reset uid information with errno '`*errno*`' - '`*errorreason*`'.`

  The execution of the command *command* failed trying to reset effective uid.

  Action:
  Depends on the errno value.

- `The configuration file "`*nondefaultconfig*`" has been specified as option argument of the -c option, but the Wizard Tools activated configuration is "`*defaultconfig*`" (see `*defaultconfig*`). The base monitor will not be started. The desired configuration file should be re-activated using the Wizard Tools hvw command.`

  This message is shown when the user tries to start the RMS with a configuration different from the configuration present in the RMS default configuration file. The base monitor is not started, the user will need to either change the default configuration file by re-activating the configuration via the Wizard Tools `hvw` command or specify the proper option argument for the `-c` option.

  Action:
  The user should correct the default configuration by activating the specified configuration file using the Wizard Tools or specify the proper option argument to the `-c` option.

- `The file '`*filename*`' could not be opened: `*errormsg*

  While performing a `hvdump`, if the file *filename* could not be opened because of *errormsg*, this message is the result and `hvdump` exits with exit code 8.

  Action:
  Take action based on the *errormsg*.

- `The length of return message from BM is illegal (`*actuallength* `actual` *expectedlength* `expected).`

  When the `hvassert` utility expecting a return message from the base monitor receives a message of length *actuallength* when it is expecting a message of length *expectedlength*, this message is printed and `hvassert` exits with exit code 5.

  Action:
  Contact field support.

- `The system call` *systemcall* `could not be executed:` *errormsg*

  While performing a `hvdump`, if the *systemcall* could not be executed because of *errormsg*, this message is the result and `hvdump` exits with exit code 7.

  Action:
  Take action based on the *errormsg*.

- `The user has invoked the hvcm command with the -a flag on a host where RMS is already running, sending request to start all remaining hosts.`

  If `hvcm` is invoked with the `-a` flag, then RMS will be started on the other hosts in the cluster.

  Action:
  None required.

- `timed out! Most likely rms on the remote host is dead.`

  While performing `hvrcp`, if the command times out because the base monitor on the local host has not received an acknowledgement from the base monitor on the remote host, the most probable reason is that the RMS on the remote host is dead.

  Action:
  Make sure that the RMS on the remote host is running.

- `Too many arguments, usage: hvmod -E`

  The `hvmod` utility does not expect any arguments when invoked with the `-E` option. If not, `hvmod` exits with exit code 1.

  Action:
  Make sure that '`hvmod -E`' is not invoked with any arguments.

- `Too many asserted objects,` ***maximum*** `is the max.`

  Any attempt to assert on a number of objects which is greater than the *maximum* will cause this message to be printed.

  Action:
  Make sure that the number of asserted objects is less than the max.

- `Usage: hvassert [-h SysNode] [-q] -s resource_name resource_state | [-h SysNode] [-q] -w resource_name resource_state seconds`

  If the utility `hvassert` has been invoked in a way that does not conform to its expected usage, this message is printed and the utility exits with exit code 6.

  Action:
  Follow the usage specified above.

- `Usage: hvcm [-V] [-a] [-s targethost] [-c config_file] [-m] [-h time] [-l level] [-r count] [-w time]`

  Usage is not correct.

  Action:
  Check the `hvcm` man page for correct usage.

- `Usage: hvconfig -l | -o config_file`

  An attempt to use the `hvconfig` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvdisp {-a | -c | -h | -i | -l | -n | -S resource_name [-u | -c] | -z resource_name | -T resource_type [-u | -c] | -u | resource_name | ENV | ENVL} [-o out_file]`

  An attempt to use the `hvdisp` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvdump {-g | -f out_file | -t wait_time}`

  An attempt to use the `hvdump` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hveject -s host`

  An attempt to use the `hveject` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code of 2 or 6 depending on one of the following conditions:
  – If an unknown option is used, the exit code is 2.
  – If the `hveject` utility is invoked directly without any options or arguments, the exit code is 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvjoin -s host`

  An attempt to use the `hvjoin` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code of 2 or 6 depending on one of the following conditions:
  – If an unknown option is used, the exit code is 2.
  – If the `hveject` utility is invoked directly without any options or arguments, the exit code is 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvlogclean [-d]`

  An attempt to use the `hvlogclean` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code 6.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvmod [-i] [-l] -f config_file.us | -E | -L | [-i] [-l] -c "modification directives"`

  If the `hvmod` utility is invoked in any one of the following ways, `hvmod` exits with exit code 6:
  – If `hvmod` is invoked without any options.
  – If `hvmod` is invoked with the `-l` or `-i` options but with arguments when none are expected.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvrcp localfile node:remotefile`

  This message is the result of either of these conditions:
  – The number of arguments specified is not equal to 2.
  – The second argument is not specified in the form node:remotefile.
  `hvrcp` then exits with exit code 6.

  Action:
  Follow the intended usage of `hvrcp` as specified above.

- `Usage: hvreset [-t timeout] userApplication`

  An attempt to use the `hvreset` utility in a way that does not conform to the expected usage leads to this message and the utility exits with exit code 2.

  Action:
  Follow the expected usage for the utility.

- `Usage: hvsend { [ -m message ] [ -s system ] [ -w waittime ] dest_object | -f in_file [ dest_object ] }`

  This is a result of using an unknown option with the `hvsend` command.

  Action:
  Follow the intended usage of the utility.

- `Usage: hvshut {-f | -L | -a | -l | -s SysNode}`

  If the usage of `hvshut` does not conform to the expected usage as specified in the above message, the `hvshut` utility exits with the exit code 6.

  Action:
  Follow the usage specified above.

- `Usage: hvswitch [-f] userApplication [SysNode] | -p userApplication`

  If an unknown option is used with the `hvswitch` utility or if there are more than 2 arguments specified for `hvswitch`, it exits with exit code 6.

  Action:
  Follow the intended usage of the utility.

- `Usage: hvutil {-a | -d | -f | -c | -s} userApplication | {-t n | -N string } resource | -L{0|1} resource | {-o | -u} SysNode | -l level | -w | -W | -i {all | userApplication} | -r | -m {on|off|forceoff} userApplication | -M {on|off|forceoff}`

  This message could appear in any one of the following situations:
  - '`hvutil -u`' is invoked with more than 1 argument. Exit code 7.
  - `hvutil` is invoked without any options or arguments. Exit code 7.
  - `hvutil` is invoked with an illegal option. Exit code 7.
  - '`hvutil -i`' is used without an argument. Exit code 13.
  - '`hvutil -r`' is used with an argument. Exit code 14.
  - '`hvutil {-w | -W}`' is used with an argument. Exit code 9.
  - '`hvutil -n`' is invoked with `NoConfirm` as the only argument. Exit code 5.
  - '`hvutil {-m | -M}`' is invoked with an argument other than `on`, `off`, or `forceoff`. Exit code 16.
  - '`hvutil -m`' is invoked without an argument, or '`hvutil -M`' is invoked with an argument. Exit code 16.

Action:

Follow the intended usage of `hvutil`.

11 Appendix—Operating system error numbers

Some RMS error messages display the operating system error number, <*errno*>, that was returned when a process such as a detector or script failed. These error numbers may provide important clues in diagnosing the problem. See user document or header files provided with the relevant operating system.

12 Appendix—Object types

Table 10 contains a list of all object types that are supplied with RMS. The middle column lists the attributes that must be specified or are recommended for the object type in the object configuration file definition.

Type	Required Attributes	Description
`andOp`	`HostName` for direct children of a `userApplication` object	Object that is associated with its children by a logical `AND` operator. Define this type of object if all children have to be online or offline at the same time.
`controller`	`Resource`; either `Follow` or `Scalable`	Object within a `userApplication` that controls one or more `userApplication` objects
`gResource`	`rKind`, `rName`	Custom (generic) object
`ENV`	None required	Object containing clusterwide (global) environment variables
`ENVL`	None required	Object containing node-specific (local) environment variables
`orOp`	None required	Object associated with its children by a logical `OR` operator. Define this type of object if at least one child has to be online at all times.
`SysNode`	None required	Node object; required. Only type `userApplication` can be defined for the children.
`userApplication`	None required	User application; required for every application. Only `SysNode` parents are allowed. The attribute `HostName` must be set for children.

Table 10: Object types

13 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 PCS or the RMS Wizards.

13.1 Attributes available to the user

Attributes in this section can be changed through the PCS Wizards Tools user interface or the `hvattr` command.

- `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.

- `ApplicationSequence`

  Possible Values: Valid string (character) of the format *group1*[:*group2*[: ...]], where each group is a space-delimited list of `userApplication` object names
  Default: “” (empty)

  Valid for `Scalable controller` objects. Specifies the list of all child applications for the sequencing of `Online` or `Offline` requests. Groups separated by colons are processed sequentially from left to right for `Online` requests, and from right to left for `Offline` requests. Each group can be a single application name or a list of application names separated by spaces or tabs. All applications in a single group are processed in parallel.

  For example, if the sequence is

`app1:app2a app2b:app3`
then an `Online` request would first process `app1`, then `app2a` and `app2b` in parallel, and finally `app3`.

- **`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.

  This attribute must be `0` for `Scalable` controllers: RMS handles switchover of `Scalable` child applications automatically.

- **`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.

- **`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 (“I”) 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 | Resource-Failure`.

- **`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.

- **`Follow`**

  Possible Values: `0`, `1`
  Default: `0`

  Valid for `controller` objects. Specifies whether or not the object is a `Follow` controller. The user changes this attribute indirectly by selecting the controller type in the configuration interface.

  If set to `1`, the controller operates in `Follow` mode. When the parent application is switched `Online`, then all child applications also come `Online` on the same node, regardless of the order specified in their respective `PriorityList` attributes. Each child application must be able to run on the same set of nodes as its parent application: the controller keeps track of, and sends requests to, its child applications that are running on the same node.

  Other attributes of the controller object must be set as follows (these are set automatically by the configuration tools):

  ```
  IgnoreOfflineRequest=0
  IgnoreOnlineRequest=0
  Scalable=0
  ```

  If `Follow` is set to `1`, `Scalable` must be set to `0`. `Follow` and `Scalable` control policies are mutually exclusive.

- **`Halt`**

  Possible Values: `0`, `1`
  Default: `0`

  Valid for `userApplication` objects. Eliminates a node if a double fault occurs.

- **`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 `userAppli-cation` 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. In case of `AutoStartUp` or a priority switch, this last-`Online` node has the highest priority, regardless of its position in the priority list.
  If set to `1`, the application comes `Online` on the node where it was last `Online`.
  If not set (`0`), the application comes `Online` on the node with the highest priority in the attribute `PriorityList`.

  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 currently online nodes, even if some other nodes (including the application's primary node) are offline or faulted.

  For an application that contains a `Scalable` controller (i.e. for a parent application) `PartialCluster` must be set to `1`. Each child application must have its attributes set as follows: `PartialCluster` must be set to `0`; `AutoStartUp` must be set to `0`.

- **`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.

- `Scalable`

  Possible Values: `0`, `1`
  Default: `0`

  Valid for `controller` objects. Specifies whether or not the object is a `Scalable` controller. The user changes this attribute indirectly by selecting the controller type in the configuration interface.

If set to `1`, then the object is a `Scalable` controller and `Resource` must contain the list of child applications. Other attributes must be set as follows (these are set automatically by the configuration tools):

```
IndependentSwitch=1
AutoRecover=0
IgnoreOfflineRequest=0
IgnoreOnlineRequest=0
Follow=0
```

If `Scalable` is set to `1`, then `Follow` must be set to `0`. `Scalable` and `Follow` control policies are mutually exclusive.

- **`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. 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 that is used 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.

- **`StandbyTimeout`**

  Possible Values: `0`–*MAXINT* (in seconds)
  Default: `0`

  Valid for `controller` objects. The number of seconds to wait before reporting a state change after the child application transitions out of `Standby` state. If `Follow` is set to `1`, `StandbyTimeout` value must be set to `0`.

  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`
  Default: “” (empty)

  Valid for `userApplication` objects. The value specified determines the standby transitions that are to be executed.

  `StartUp` means that at startup, the application is requested to go to the `Standby` state, unless it is already `Online` or unless it is forced to go `Online` due to the `AutoStartUp` attribute.

  `SwitchRequest` means that after application switchover, the application that was `Online` before the switchover will transition to the `Standby` state.

  `ClearFaultRequest` means that the application is requested to go to the `Standby` state after a `Faulted` state was cleared with ‘`hvutil -c`’.

- **`StateChangeScript`**

  Possible Values: Valid script (character)
  Default: “” (empty)

  Valid for `Scalable controller` objects. Specifies the script to be executed upon state transitions of the either the child applications or the `SysNode` objects where the child applications can run. The script is executed once each time a child application transitions into one of the states `Online`, `Offline`, `Faulted`, or `Standby`, even if the transition request originates from the controller itself. The script is also executed once each time a `SysNode` in the child application object's `PriorityList` changes its state to `Offline` or `Faulted`.

- **`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`.

13.2 Attributes managed by configuration wizards

Attributes in this section are managed internally by the configuration wizards.

- `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 as defined in “Appendix—Object types”

  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 `Follow` 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.

- `IgnoreOfflineRequest`

  Possible Values: `0`, `1`
  Default: `1`

  Valid for `controller` objects. If set to `1`, then neither `PreOffline` nor `Offline` requests will be propagated to child applications. If `0`, then requests will be propagated.

  Must be `1` for a `Follow` controller. Must be `0` for a `Scalable` controller.

- `IgnoreOnlineRequest`

  Possible Values: `0`, `1`
  Default: `1`

  Valid for `controller` objects. If set to `1`, then neither `PreOnline` nor `Online` requests will be propagated to child applications. If `0`, then `Online` requests will be propagated.

  Must be `1` for a `Follow` controller. Must be `0` for a `Scalable` controller.

- `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`.

- `IndependentSwitch`

  Possible Values: `0`, `1`
  Default: `0`

  Valid for controller objects. Determines the action of controlled (child) applications when the controlling (parent) application is switched from one node to another.

  If `0`, the parent controller propagetes the switch request to each child application. A child application may not be switched on its own.

  If `1`, the parent application can be switched to another node without causing a similar switchover for the child applications. Each child application will remain online on the same node where it was running before the switchover.

  The `IndependentSwitch` attribute is ignored when the parent is switched via a forced request such as `hvswitch -f`. In this case, the controller propagates the switch request to each child application as if the attribute were set to `0`.

  Must be `0` for a `Follow` controller. Must be `1` for a `Scalable` controller.

- `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

- `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.

- `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. Also, if `0`, then only `Offline` and `Online` requests will be propagated if `IgnoreOf-flineRequest` and `IgnoreOnlineRequest` respectively are set to `0`.

14 Appendix—Environment variables

You can change the RMS environment by modifying the appropriate entries in the `hvenv.local` file and restarting RMS.

Caution

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

Refer to the *Reliant Monitor Services (RMS) Configuration and Administration Guide* for more information about environment variables.

14.1 Global environment variables

Global variable settings (`ENV`) are included in the configurations checksum that is common to the cluster. The checksum is verified on each node during startup of the base monitor. RMS will fail to start if it detects a checksum difference between the values on any two nodes.

The default values of the environment variables are found in *<RELIANT_PATH>*`/bin/hvenv`. They can be redefined in the `hvenv.local` command 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 `SysNode`s 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 this environment variable is used, ensure that it is correctly defined on all cluster nodes and that it is always kept up-to-date. When a node is brought back into the cluster, remove it from this environment variable. If this does not occur, data loss could occur because RMS will ignore this node during the startup procedure and will not check whether the application is already running on the nodes specified in this list. It is the system administrator's responsibility to keep this list up-to-date if it is used.

- **`HV_AUTOSTART_WAIT`**

  Possible values: `0`–*MAXINT*
  Default: `60` (seconds)

  Defines the period (in seconds) that RMS waits for cluster nodes to report `Online` when RMS is started. If this period expires and not all cluster nodes are online, a switchlog message indicates the cluster nodes that have not reported `Online` and why the user application(s) cannot be started automatically.

  This attribute generates a warning message only. `AutoStartUp` will proceed even if the specified period has expired.

- **`HV_CHECKSUM_INTERVAL`**

  Possible values: `0`–*MAXINT*
  Default: `120` (seconds)

  Interval in seconds for which the RMS base monitor waits for each `Online` node to verify that its checksum is the same as the local checksum.

  If checksums are confirmed within this interval, then RMS on the local node continues its operations as usual. However, if a checksum from a remote node is not confirmed, or if it is confirmed to be different, then the local monitor shuts down if it has been started less than `HV_CHECKSUM_INTERVAL` seconds before.

Also, if a checksum from a remote node is not confirmed, or if the checksum is confirmed to be different, then the local monitor considers the remote node as `Offline` if that local monitor has been started more than `HV_CHECKSUM_INTERVAL` seconds before.

- **`HV_LOG_ACTION_THRESHOLD`**

  Possible values: `0–100`
  Default: `98`

  Defines the behavior of `hvlogcontrol`. 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 345 for more information on `HV_LOG_ACTION`.

- **`HV_LOG_WARN_THRESHOLD`**

  Possible values: `0–100`
  Default: `95`

  Defines the behavior of `hvlogcontrol`. If the used space on the file system containing the RMS log disk is larger or equal to this threshold value, the `hvlogcontrol` 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 and RMS wizard 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: `150` (seconds)

  Defines the period (in seconds) that the command `hvshut` waits before timing out and generating an error message. This variable must be set to the maximum value required to successfully terminate offline processing for a specific application. This value corresponds to the maximum time required by an application to go offline (on all cluster nodes if the `–a hvshut` option is used).

  If this value is too low, the `hvshut` command will time out and generate an error message. However, this does not mean that the shutdown process is stopped; it merely means that the `hvshut` command itself will time out. The shutdown process continues within the RMS system. This means that the system shuts down successfully after an `hvshut` command has timed out, even though the command has exited.

14.2 Local environment variables

i The default values of the environment variables are found in *<RELIANT_PATH>*`/bin/hvenv`. They can be redefined in the `hvenv.local` command file.

The following list describes the local environment variables for RMS:

- `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 node detector `hvdet_node` uses for connections to all remote cluster nodes before assuming that the connection attempt has failed.

- **`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_RCSTART`**

  Possible values: `0`, `1`
  Default: `1` (start RMS in the `rc` script)

  Determines if RMS is started in the `rc` script. (Prerequisite for `rc` start: `CONFIG.rms` exists and contains a valid entry.)

- **`HV_REALTIMEPRIORITY`**

  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.

By default, the base monitor and detectors are real time processes. However, if the base monitor has been started with the `-R` non-real-time flag, the value of `HV_REALTIME_PRIRORITY` is disregarded.

- **`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 variable is not set by default. 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, and is not run once for each user application or node.

- `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.

15 Appendix—List of manual pages

This appendix lists the online manual pages for CCBR, CF, CFS, CIP, Monitoring Agent, PAS, PCS, RCVM, 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
```

15.1 CCBR

System administration

`cfbackup`
: save the cluster configuration information for a PRIMECLUSTER node

`cfrestore`
: restore saved cluster configuration formation on a PRIMECLUSTER node

15.2 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

`cftool`
: print node communications status for a node or the cluster

15.3 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

15.4 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

15.5 Monitoring Agent

System administration

`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

15.6 PAS

System administration

`mipcstat`
: MIPC statistics

`clmstat`
: CLM statistics

15.7 PCS

System administration

`pcstool`
: Modifies PCS configurations from the command line

`pcscui`
: Character-based interface for PCS

`pcs_reinstall`
: Utility for re-integrating PCS with dependent products

`maketrusted`
: Utility to install signed version of PCS

15.8 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

15.9 Resource Database

i To display a Resource Database manual page, add `/etc/opt/FJSVcluster/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

`clstartrsc`
: resource activation

`clstoprsc`
: resource deactivation

`clsyncfile`
: distribute a file between cluster nodes

User command

`clgettree`
: display the tree information of the resource database

15.10 RMS

System administration

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

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

`hvcm`
: start the RMS configuration monitor

`hvconfig`
: display or save the RMS configuration file

`hvdisp`
: display RMS resource information

`hvdist`
: distribute RMS configuration files

`hvdump`
: collect debugging information about RMS

`hvgdmake`
: compile an RMS custom detector

`hvlogclean`
: clean RMS log files

`hvrclev`
: change default RMS start run level

`hvreset`
: reinitialize the graph of an RMS user application

`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

15.11 RMS Wizards

RMS Wizard Tools and RMS Wizard Kit

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

<*RELIANT_PATH*>`/htdocs./wizards.en` (Solaris)

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

The default value of <*RELIANT_PATH*> is `/opt/SMAW/SMAWRrms/`.

15.12 SCON

`scon`
: start the cluster console software

15.13 SF

System administration

`rcsd`
: Shutdown Daemon of the Shutdown Facility

`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

`sdtool`
: interface tool for the Shutdown Daemon

15.14 SIS

System administration

`dtcpadmin`
: start the SIS administration utility

`dtcpd`
: start the SIS daemon for configuring VIPs

`dtcpstat`
: status information about SIS

15.15 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 PRIMECLUSTER 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 RMS Wizards, 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 the RMS 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)*.

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*.

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 PRIME-CLUSTER 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 RMS Wizards, 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)*.

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 s 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*.

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)*.

primary node (RMS)
The default node on which a user application comes online when RMS is started. This is always the nodename 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*.

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 node
A node that is accessed through a LAN or telecommunications line.

See also *local node, node*.

reporting message (RMS)
A message that a detector uses to report the state of a particular resource to the base monitor.

resource (RMS)
A hardware or software element (private or shared) that provides a function, such as a mirrored disk, mirrored disk pieces, or a database server. A local resource is monitored only by the local node.

See also *private resource (RMS)*, *shared resource*.

resource definition (RMS)
See *object definition (RMS)*.

resource label (RMS)
The name of the resource as displayed in a system graph.

resource state (RMS)
Current state of a resource.

RMS
See *Reliant Monitor Services (RMS)*.

RMS commands
Commands that enable RMS resources to be administered from the command line.

RMS configuration
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 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 Wizard Tools or PCS interface to provide controls for the additional features.

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

RMS Wizard Tools

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*, *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)*.

shared resource

A resource, such as a disk drive, that is accessible to more than one node.

See also *private resource (RMS)*, *resource (RMS)*.

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), 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 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)*, *striped virtual disk (RCVM)*.

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.

Wizard Kit (RMS)
See *RMS Wizard Kit*.

Wizard Tools (RMS)
See *RMS Wizard Tools*.

Abbreviations

AC	Access Client
API	application program interface
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
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
NIC	network interface card
NSM	Node State Monitor

OSD
operating system dependent

PAS
Parallel Application Services

PCS
PRIMECLUSTER Configuration Services

RCCU
Remote Console Control 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

SD
Shutdown Daemon

SF
Shutdown Facility

SIS
Scalable Internet Services

VIP

Virtual Interface Provider

Figures

Figure 1: Overview of PRIMECLUSTER 10

Figure 2: Controlled application scenario 12

Figure 3: RMS representation of controlled application 13

Figure 4: Follow mode switchover 14

Figure 5: Scalable mode controlled (child) application switchover . 15

Figure 6: Scalable mode controlling (parent) application switchover 15

Figure 7: Relationship between RMS and RMS Wizards 18

Figure 8: NFS Lock Failover screen 38

Figure 9: Main configuration menu when RMS is not active 42

Figure 10: Main configuration menu when RMS is running 46

Figure 11: Application type selection 47

Figure 12: Menu leading to basic settings 48

Figure 13: Menu to configure basic settings 48

Figure 14: Menu to configure non-basic settings 49

Figure 15: Main configuration menu 50

Figure 16: Activating a configuration 51

Figure 17: Quitting the Main configuration menu 52

Figure 18: Main configuration menu 58

Figure 19: Add hosts to a cluster menu 60

Figure 20: Remove hosts from a cluster menu 61

Figure 21: Main configuration menu 61

Figure 22: Application type selection menu 62

Figure 23: Prompting for further actions 63

Figure 24: Consistency check and Machines+Basics menu 64

Figure 25: List of nodes for failover procedure 65

Figure 26: Machines+Basics menu for additional nodes 65

Figure 27: AutoSwitchOver mode 66

Figure 28: Setting flags for AutoSwitchOver mode 66

Figure 29: Saving settings 67

Figure 30: Non-basic settings 68

Figure 31: Prompting for display specification 69

Figure 32: List of display options 70

Figure 33: Successful consistency check for APP1 71

Figure 34: Turnkey wizard DEMO 72

Figure 35: Global settings: main menu 73

Figure 36: Global settings: machines menu 74

Figure 37: Global settings: AlternateIps first menu 74

Figure 38: Global settings: AlternateIps second menu 74

Figure 39: Global settings: AlternateIps first menu with first interface 75

Figure 40: Global settings: AlternateIps first menu with both interfaces 75

Figure 41: Global settings: main menu with AlternateIps for first host 75

Figure 42: Global settings: main menu with AlternateIps for both hosts 76

Figure 43: Main configuration menu 77

Figure 44: Successful configuration activation 77

Figure 45: Quitting the Main configuration menu 78

Figure 46: Starting again with the Main configuration menu 79

Figure 47: Application type selection menu 80

Figure 48: Prompting for further specification 81

Figure 49: Machines+Basics menu 81

Figure 50: List of nodes for failover procedure 82

Figure 51: Machines+Basics menu 82
Figure 52: Non-basic settings 83
Figure 53: Assigning a controller 84
Figure 54: List of applications to be chosen as controlled applications 84
Figure 55: Menu for setting controller flags 85
Figure 56: Changing controller timeout period 85
Figure 57: Saving flags for controller 86
Figure 58: Indication of flags set for controller 86
Figure 59: Menu with settings for GENERIC turnkey wizard 87
Figure 60: Main configuration menu 87
Figure 61: Main configuration menu 88
Figure 62: Activating the configuration for the second time 88
Figure 63: Return to Main configuration menu 89
Figure 64: Invoking the Cluster Admin GUI 92
Figure 65: Web-Based Admin View login screen 93
Figure 66: Top menu 94
Figure 67: Cluster menu 95
Figure 68: Main screen 96
Figure 69: RMS main window 97
Figure 70: RMS tree with a controller object 98
Figure 71: Configuration information or object attributes 99
Figure 72: Command pop-up 100
Figure 73: Command pop-up for an offline application 101
Figure 74: Confirmation pop-up window 102
Figure 75: Confirmation pop-up window for scalable application . . 102
Figure 76: Viewing the RMS switchlog file 103

Figure 77: Viewing the RMS switchlog file in a detached window . . 104
Figure 78: Viewing the application log 105
Figure 79: Search based on date and time filter 106
Figure 80: Using the Find pop-up in log viewer 107
Figure 81: RMS full graph 109
Figure 82: RMS application graph 110
Figure 83: RMS subapplication graph 111
Figure 84: Composite subapplication graph 112
Figure 85: Configuration information pop-up 113
Figure 86: Command pop-up 114
Figure 87: RMS graph with affiliation names 115
Figure 88: RMS graph with resource names 116
Figure 89: RMS graph with affiliation names and resource names . 117
Figure 90: RMS graph after RMS is shut down 118
Figure 91: Clusterwide table 119
Figure 92: Faulted and offline applications in the clusterwide table . 120
Figure 93: Exclamation marks in clusterwide table and the RMS tree 121
Figure 94: Command pop-ups in clusterwide table 122
Figure 95: Before RMS is shut down 123
Figure 96: After RMS is restarted with a different configuration . . . 124
Figure 97: Starting RMS from the main menu 126
Figure 98: RMS Start Menu for all nodes 127
Figure 99: RMS Start Menu for individual nodes 128
Figure 100: Starting RMS on individual nodes 129
Figure 101: Stopping RMS 130

Figure 102: Stopping RMS on all available nodes 131
Figure 103: Stopping RMS on one node from the list 132
Figure 104: Using command pop-up to stop RMS 133
Figure 105: Starting an application 134
Figure 106: Switching an application 136
Figure 107: Switching a busy application 137
Figure 108: Shutting down an application 138
Figure 109: Clearing an application fault 140
Figure 110: Clusterwide environment variables 142
Figure 111: Local environmental variables pop-up 143
Figure 112: Local environmental variables window 144
Figure 113: Displaying application states 145
Figure 114: Invoking the log viewer 174
Figure 115: Find pop-up window 175
Figure 116: Detached log 176
Figure 117: Resource-based search 177
Figure 118: Results of time-based search 178
Figure 119: Results of keyword-based search 179
Figure 120: Results of severity-level-based search 181
Figure 121: Controlling the log level with PCS 184

Tables

Table 1: Available CLI commands 24
Table 2: RMS base directory structure 29
Table 3: Log directory structure 30
Table 4: RMS host name conventions in /etc/hosts 35
Table 5: Cluster site planning worksheet 59
Table 6: Switch processing activities 166
Table 7: Log files 172
Table 8: Descriptions of severity levels 180
Table 9: Log levels 182
Table 10: Object types 323

Index

>> input prompt 41

A

activating
- application 138
- configuration 44, 49, 77
- configuration second time 88

administrative privileges 93
Affiliation, attribute 335
alternate interfaces 35, 73
AlternateIp, attribute 325
AlternateIps 35, 73
andOp
- attributes 323
- description 323

application
- switching to SysNode 23, 25

application graph 110
application logs
- displaying 105
- files 103
- searching text 107

applications
- activating 138
- as objects 11
- dependencies 98
- displaying states 145
- going offline 157
- switching over 165
- taking offline 138
- viewing logs 146

ApplicationSequence
- attribute 325

attributes
- Affiliation 335
- AlternateIp 325
- ApplicationSequence 325
- AutoRecover 326
- AutoRecoverCleanup 335
- AutoStartUp 326
- AutoSwitchOver 326
- Class 335
- ClusterExclusive 326
- Comment 335
- ControlledShutdown 336
- DetectorStartScript 336
- FaultScript 327
- Follow 327
- Halt 328
- HostName 336
- I_List 328
- IgnoreOfflineRequest 336
- IgnoreOnlineRequest 337
- IgnoreStandbyRequest 337
- IndependentSwitch 337
- LieOffline 338
- MaxControllers 328
- MonitorOnly 328
- NoDisplay 338
- NullDetector 338
- OfflineDoneScript 338
- OfflineScript 328
- OnlinePriority 329
- OnlineScript 329
- OnlineTimeout 338
- PartialCluster 330
- PersistentFault 339
- PostOfflineScript 330
- PostOnlineScript 330
- PreCheckScript 339
- PreOfflineScript 330
- PreOnlineScript 331
- PreserveState 331
- PriorityList 331
- Resource 339
- rName 339
- Scalable 331
- ScriptTimeout 332
- ShutdownPriority 332
- SplitRequest 340
- StandbyCapable 332

StandbyTimeout 333
StateChangeScript 333
WarningScript 334
AutoRecover
fault processing 162
AutoRecover, attribute 326
AutoRecoverCleanup, attribute 335
AutoStartUp, attribute 326
AutoSwitchOver
fault processing 160
AutoSwitchOver, attribute 326

B
base monitor 20
debug messages 171
detectors 54
high availability 11
log file 172
log levels 182
messages 172
stack tracing 182
switchlog 186
basic settings, wizards 47
bin, directory 29
bmlog description 172
browser 91
build, directory 29

C
CF commands
cfconfig 349
cftool 349
cfset 349
cfsmntd 350
CIP commands
cip.cf 350
cipconfig 350
ciptool 350
Class, attribute 335
clbackuprdb 352
clearing
faulted resources 25
Faulted state 140
faults 140
hung nodes 25
SysNode Wait state 140
clgettree 353
CLI
options 125
See RMS CLI
clinitreset 352
clrestorerdb 353
clsetparam 353
clsetup 353
clstartrsc 353
clstoprsc 353
cluster 1
high availability 10
services 9
switching user applications 23
Cluster Admin 20, 125
administrative privileges 93
application graph 110
application log files 103
clusterwide table 119
command pop-ups 100
configuration 113
graph 112
GUI 91
logging in 93
main screen 95
object attributes 99
operator privileges 93
overview 10
primary management server 92
RMS graphs 108
RMS main window 97
RMS tree 97
root privileges 93
searching log text 107
secondary management server 92
starting 91
switchlog 103
switchlog panel 104
SysNode selection 100
userApplication selection 100
using 91

cluster file system 9
Cluster Foundation
  LEFTCLUSTER 164
cluster node
  detector timeout for remote 345
  ignore at startup 341
  wait to report online 342
cluster volume management 9
ClusterExclusive, attribute 326
clusterwide table 119
Cmdline, resource wizard 33
command pop-ups
  clusterwide table 122
  RMS graph 114
  RMS tree 100
commands
  hvswitch -f 163
  hvutil -c 163
Comment, attribute 335
composite subapplication graph 112
configuration information, graphs 113
configurations
  displaying 24
  displaying information 96, 97
  general procedure 40
  graph 108
  individual node details 99
configuring
  applications 31, 33
  disk groups 33
  file systems 33
  IP addresses 33
  resources 33
console error messages 295
controlled applications 12
ControlledShutdown, attribute 336
controller
  attributes 323
  dependencies 98
  description 323
  graph 112
Controller, resource wizard 33
controllers 12
  Follow mode 13
  Scalable mode 14
creating
  application 58
  second application 79

D
Deact state 22
debug level, wizards 190
debug messages 169
  base monitor 171
  log directory 171
  severity level 180
  wizards 187, 189
debug reporting, wizards 190
DEBUG statements, wizards 190
defining timeout 345
DEMO turnkey wizard 33, 41, 61, 62
dependant resources 11
dependencies 98
det_disklog file 173
detectors 10, 11
  fault situations 156
  illegal 151
  RMS Wizard Kit 19
  RMS Wizard Tools 19
  starting 129
DetectorStartScript, attribute 336
directed switch requests 165
directories
  bin 29
  build 29
  etc 29
  include 29
  lib 29
  us 29
directory hierarchy
  root directory 344
  specifying root directory 344
disk classes, as application resources 16
displaying
  application states 145
  current RMS configuration 24
  environment variables 142

dkconfig 352
dkmigrate 352
dkmirror 352
dktab 352
documentation
  additional 2
  wizards 55

E
echo service 39
ENV
  attributes 323
  description 323
ENV and ENVL objects 27
environment variables 27
  displaying 142
  HV_AUTOSTART_WAIT 342
  HV_AUTOSTARTUP_IGNORE 341
  HV_CHECKSUM_INTERVAL 342
  HV_CONNECT_TIMEOUT 345
  HV_LOG_ACTION 346
  HV_LOG_ACTION_THRESHOLD 343
  HV_LOG_WARN_THRESHOLD 343
  HV_MAXPROC 346
  HV_RCSTART 346
  HV_REALTIMEPRIORITY 346
  HV_SCRIPTS_DEBUG 347
  HV_SYSLOG_USE 347
  HV_WAIT_CONFIG 344
  RELIANT_HOSTNAME 347
  RELIANT_INITSCRIPT 348
  RELIANT_LOG_LIFE 344
  RELIANT_LOG_PATH 171, 344
  RELIANT_PATH 344
  RELIANT_SHUT_MIN_WAIT 345
  RELIANT_STARTUP_PATH 348
  SCRIPTS_TIME_OUT 348
ENVL
  attributes 323
  description 323
error messages 169
  base monitor 171
  console 295
  fatal 281
  switchlog 195
errors
  at initialization 157
  during offline processing 158
  in offline state 162
  reaction to 159
etc, directory 29

F
failover 11
fatal error messages 281
fault clearing 163
fault script 160
Faulted state 21
  clearing 140
  FaultScript 23
faults
  clearing 140
  failover 64
FaultScript
  attribute 327
  script 23
file systems
  as application resources 16, 31
  filling up 191
  Fsystem 33
  resource type 19
  site preparation 34
  warning threshold 343
fjsvwvbs 356
fjsvwvcnf 356
Follow mode controllers 13
Follow, attribute 327
forced shutdown 132
forced switchover 164
forced-online requests 163
fsck_rcfs 350

Fsystem, resource wizard 33

G
Gds, resource wizard 33
GENERIC turnkey wizard 33, 80
Global Disk Services 33
global environment variables 27
Global Link Services 33
Gls, resource wizard 33
graph, RMS 44
graphical user interface *See* GUI
graphs 108
  application 110
  command pop-ups 114
  composite subapplication 112
  configuration information 113
  reinitializing 25
gResource 54
  object type 323
  required attributes 323
GUI 125
  after shut down 118
  messages 146
  pull-down menus 96
  starting RMS 97

H
Halt, attribute 328
high availability 1, 9
  specifying applications 43
HostName, attribute 336
hosts
  site preparation 34
HV_AUTOSTART_WAIT 342
HV_AUTOSTARTUP_IGNORE 341
HV_CHECKSUM_INTERVAL 342
HV_CONNECT_TIMEOUT 345
HV_LOG_ACTION 346
HV_LOG_ACTION_THRESHOLD 343
HV_LOG_WARN_THRESHOLD 343
HV_MAX_HVDISP_FILE_SIZE 346
HV_MAXPROC 346
HV_RCSTART 346
HV_REALTIMEPRIORITY 346
HV_SCRIPTS_DEBUG 347
HV_SYSLOG_USE 347
HV_WAIT_CONFIG 344
hvcm -l 182
hvdisp command 27, 29, 144, 145
  file size 346
  no display 338
hvenv and hvenvl.local files 27
  changing variables 28
hvexec command 53
hvreset, defined 25
hvshut
  defined 25
hvshut command
  defining timeout 345
  stopping RMS 133
hvswitch
  defined 25
hvswitch command
  userApplication 135
hvswitch -f 163
hvutil 165
hvutil -c 163
hvutil command
  defined 25
  shutting down an application 138
hvw command 19
  defined 40
  operation mode 45
  resuming configuration 79

I
I_List, attribute 328
IgnoreOfflineRequest, attribute 336
IgnoreOnlineRequest, attribute 337
IgnoreStandbyRequest, attribute 337
include, directory 29
Inconsistent state 22
IndependentSwitch, attribute 337
initial state
  initializing 151
  Unknown state 151
initialization script, specifying 348

initialization, error at 157
InitScript 22
interfaces, alternate 35, 73
IP addresses
  defining resources 16
  resource wizard 33
Ipaddress, resource wizard 33

K
killing a node 12

L
LAN interfaces 36
LEFTCLUSTER 164
lib, directory 29
LieOffline, attribute 338
local environment variables 27
log files
  application 103
  base monitor 172
  interpreting 185
  node detector 173
  specify directory 344
  switchlog 165, 173
  time of preservation 344
  viewing 146
log levels, specifying 182
log messages, wizards 187
logging in, Cluster Admin 93

M
MA commands
  clrccumonctl 351
  clrccusetup 351
  clrcimonctl 351
main menu
  wizards 42
maketrusted 351
making forced-online requests 163
management server 92
manual pages
  display 349
  listing 349
market-specific applications 10
MaxControllers, attribute 328
messages 149
  base monitor 172
  bmlog 172
  debug 171, 180
  error 171
  generic detector log 173
  node detector 173
  troubleshooting RMS 191
  wizards 187
messages, error
  console 295
  fatal 281
  switchlog 195
MonitorOnly, attribute 328
mount_rcfs 350

N
naming conventions, RMS 35, 59
networks
  site preparation 34
ngadmin 350
node
  killing 12
node detector log files 173
node names in configuration files 34, 35
nodes 11
NoDisplay, attribute 338
non-basic settings, wizards 47
NullDetector, attribute 338

O
object types
  andOp 323
  controller 323
  ENV 323
  ENVL 323
  gResource 323
  orOp 323
  SysNode 323
  userApplication 323
objects

activating applications 139
attributes 26, 99
clearing a fault 140
clusterwide table 119
command pop-ups 114, 122
Controller 12
controller 98, 112
dependencies 108
graph customization 115
information 113
relationships 108
RMS full graph 108
RMS tree 100
selecting 100
starting an application 134
switching applications 136
SysNode 11, 100
taking application offline 138
types 26
userApplication 11, 100, 133
offline
request 157
offline processing 11
definition 157
fault situations 158
offline scripts 19
Offline state 21
OfflineDoneScript 23
OfflineScript 23
PostOfflineScript 23
offline state 11
OfflineDoneScript
attribute 338
script 23
OfflineFault state 21
OfflineScript
script 23
OfflineScript, attribute 328
online processing 11
online scripts 19
Online state 21
OnlineScript 23
PostOnlineScript 23
PreCheckScript 22
PreOnlineScript 23
online state 11
OnlinePriority, attribute 329
OnlineScript
attribute 329
script 23
OnlineTimeout, attribute 338
operator intervention 165
operator privileges 93
orOp, object type 323

P
parallel application support 9
PartialCluster, attribute 330
PAS commands
clmtest 351
mipcstat 351
pcs_reinstall 351
pcscui 351
pcstool 351
PersistentFault, attribute 339
physical disks
state at initialization 156
PostOfflineScript
attribute 330
script 23
PostOnlineScript
attribute 330
script 23
PreCheckScript
attribute 339
script 22
PreOffline processing 158
PreOfflineScript
attribute 330
PreOnlineScript
attribute 331
script 23
PreserveState
effect on fault processing 160, 161
PreserveState, attribute 331

primary management server 92
PRIMECLUSTER 9
priority switch
  request 165
PriorityList, attribute 331
privileges 93
procedures 125

R
rcfs_fumount 350
rcfs_list 350
rcfs_switch 350
rcsd 355
rcsd.cfg 355
Rcvm, resource wizard 33
Reliant Monitor Services
  clusterwide table 119
  components 20
  full graph 108
  graphs 108
  high availability 10
  main window 97
  overview 10
  tree 97
RELIANT_HOSTNAME 347
RELIANT_INITSCRIPT 348
RELIANT_LOG_LIFE 344
RELIANT_LOG_PATH 171, 344
RELIANT_PATH 344
RELIANT_SHUT_MIN_WAIT 345
RELIANT_STARTUP_PATH 348
request 149
  blocking 167
  offline 157
request-triggered scripts
  InitScript 22
  OfflineScript 23
  OnlineScript 23
  PreCheckScript 22
  PreOnlineScript 23
resource wizards
  Cmdline 33
  Controller 33
  Fsystem 33
  Gds 33
  Gls 33
  Ipaddress 33
  Rcvm 33
  Vxvm 33
Resource, attribute 339
resources
  clearing faulted 25
  configuring 33
  defining 16
  dependant 11
  executing scripts 45
  file system entries 36, 38
  LAN interfaces 36
  monitoring 54
  non-basic settings 49
  object types 26
  scripts 21
  shared remote entries 37
  states 10
rKind 339
RMS
  graph 44
  naming conventions 35, 59
RMS CLI 23
  hvassert 24
  hvattr 24
  hvcm 24
  hvconfig 24
  hvdisp 24
  hvdist 24
  hvdump 24
  hvgdmake 24
  hvlogclean 25
  hvrclev 25
  hvreset 25
  hvshut 25
  hvswitch 25
  hvthrottle 25
  hvutil 25
  switching userApplication 23
RMS commands
  hvassert 353
  hvcm 24, 129, 353

hvconfig 24, 353
hvdisp 24, 145, 353
hvdist 24, 353
hvdump 24, 354
hvenv.local 354
hvexec 53
hvgdmake 24, 354
hvlogclean 25, 354
hvrclev 25
hvreset 25, 354
hvshut 25, 354
hvshut command 133
hvswitch 25, 135, 354
hvthrottle 25, 354
hvutil 25, 138, 354
RMS Wizard Kit 17, 19
detectors 19
hvw command 19
overview 10
scripts 19
RMS Wizard Tools 17
detectors 19
hvw command 19
overview 10
resource types 19
scripts 19
RMS Wizards *See* wizards 61
RMS *See* Reliant Monitor Services
rName, attribute 339
root privileges 93
running processes 16

S
SA_blade.cfg 355
SA_rccu.cfg 355
SA_rps.cfg 355
SA_rsb.cfg 355
SA_scon.cfg 355
SA_sspint.cfg 355
SA_sunF.cfg 355
SA_wtinps.cfg 355
scalability 9
scalable controller
state change script 23
Scalable mode controllers 14
Scalable, attribute 331
SCON
reply time 165
scon 355
script time-out 165
scripts 11, 22
allocating 150
Offline 19
Online 19
resources 21
RMS Wizard Kit 19
timeout 348
SCRIPTS_TIME_OUT 348
ScriptTimeout, attribute 332
sdtool 355
searching log text 107
secondary management server 92
secondary menus, wizards 46
send clear-fault request 163
severity levels
Alert 180
Critical 180
Debug 180
Emergency 180
Error 180
Info 180
Notice 180
Warning 180
Shutdown Facility 12
ShutdownPriority, attribute 332
SIS commands
dtcpadmin 356
dtcpd 356
dtcpdbg 356
site preparation 34
software monitor
function 1
RMS 10
SplitRequest, attribute 340
Standby state 21
StandbyCapable, attribute 332
StandbyTimeout, attribute 333

starting
  an application 134
  RMS 126
startup file 130
state changes
  nodes 149
StateChangeScript
  script 23
StateChangeScript, attribute 333
states 21
  Deact 22
  displaying information 119
  Faulted 21
  Inconsistent 22
  Offline 21
  OfflineFault 21
  Online 21
  Standby 21
  Unknown 22
  Wait 22, 140
  Warning 21
state-triggered scripts
  FaultScript 23
  OfflineDoneScript 23
  PostOfflineScript 23
  PostOnlineScript 23
  WarningScript 23
stopping RMS 130
strace, Linux trace tool 194
subapplication graph 111
subapplications 98
sub-menus, wizards 46
summary table 119
switch processing
  definition 165
  fault situations 167
switching
  application to Sysnode 23, 25
switching an application 136
switchlog 165
  file 171, 173
  panel 104
  viewing 103, 145
switchlog error messages 195
SysNode 11, 54
  detector 164
  fault 164
  initializing 151
  object selection 100
  object type 323
  switching application to 23, 25
  Wait state, clearing 140
Sysnode
  state change script 23
system files, and site preparation 34

T
tables
  clusterwide 119
  command pop-ups 122
taking an application offline 138
truss, Solaris trace tool 194
turning off wizard debug output 189
turnkey wizards 32, 43, 55
  DEMO 41, 61
  GENERIC 80
  ORACLE 33
  R/3 33
turnkey wizards *See also* wizards 61

U
Unknown state 22
  exiting 151
  initial state 151
us, directory 29
userApplication 54
  activating 138
  clearing fault 140
  hvswitch command 135
  object 11
  object selection 100
  object type 323
  RMS tree 97
  state change script 23
  state information 119
  taking Offline 138
  with hvshut 133
userApplication node

initializing 151

V

vdisk 352
viewing
- application logs 146
- GUI message 146
- switchlogs 145

volume managers 1, 16
Vxvm, resource wizard 33

W

Wait state 22, 165
- clearing faulted resources 25
- clearing hung nodes 25
- clearing SysNode 140

Warning state 21
- WarningScript 23

WarningScript 23
WarningScript, attribute 334
Web-Based Admin View
- login 93
- primary management server 92
- secondary management server 92

Wizard Kit 17, 19
- configuration 32
- overview 10

Wizard Tools 32
wizards
- basic settings 47
- configuring 16, 17, 32
- debug level 190
- debug messages 189
- debug reporting 190
- DEBUG statements 190
- DEMO turnkey 41
- frequently used items 41
- general description 32
- GENERIC turnkey 80
- hvexec command 53
- main menu 42
- non-basic settings 47
- ORACLE 33
- R/3 33
- resource wizards 33
- secondary menus 46
- sub-menus 46
- turn off debug output 189
- turnkey 32, 43, 55

wizards log messages 187
wvCntl 356
wvGetparam 356
wvSetparam 356
wvstat 356

Fujitsu Siemens Computers GmbH
User Documentation
33094 Paderborn
Germany

Fax: (++49) 700 / 372 00001

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

Comments
Suggestions
Corrections

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

Fax: (++49) 700 / 372 00001

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

Comments
Suggestions
Corrections

Submitted by

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