

PRIMECLUSTER™

Reliant Monitor Services (RMS) (Oracle Solaris / Linux)
Reference Guide 4.3

Copyright and Trademarks

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

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

EMC, PowerPath, and Symmetrix are registered trademarks of EMC Corporation.

TimeFinder and SRDF are trademarks of EMC Corporation.

PRIMECLUSTER is a trademark of Fujitsu Limited.

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

Requests

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

All Rights Reserved, Copyright (C) FUJITSU LIMITED 2008-2013.

Preface

Advanced RMS concepts

Scalable controllers

Troubleshooting

Non-fatal error messages

Fatal error messages

Console error messages

Warning messages

Notice messages

Appendix—Operating system error numbers

Continued ►

Appendix—States

Appendix—Object types

Appendix—Attributes

Appendix—Environment variables

Appendix—RMS command line interface

Appendix—List of manual pages

Glossary

Abbreviations

Figures

Tables

Index



Contents

1	Preface	1
1.1	About this manual	1
1.2	PRIMECLUSTER documentation list	2
1.3	Conventions	4
1.3.1	Notation	4
1.3.1.1	Prompts	4
1.3.1.2	Manual page section numbers	4
1.3.1.3	The keyboard	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 line syntax	6
1.4	Important notes and cautions	6
1.5	Abbreviations	7
1.6	Editing record	7
2	Advanced RMS concepts	9
2.1	Internal organization	9
2.1.1	Application and resource description	9
2.1.2	Messages	10
2.2	Initializing	10
2.3	Online processing	15
2.3.1	Online request	15
2.3.1.1	Manual methods	15
2.3.1.2	Automatic methods	16
2.3.2	PreCheckScript	17
2.3.3	Online processing in a logical graph of a userApplication	18
2.3.4	Unexpected reports during online processing	20
2.3.5	Fault situations during online processing	20
2.3.6	Initialization when an application is already online	21
2.4	Offline processing	22
2.4.1	Offline request	22
2.4.2	Offline processing in a logical graph of a userApplication	23
2.4.3	Unexpected reports during offline processing	24
2.4.4	Fault situations during offline processing	25
2.4.5	Object is already in Offline state	25
2.4.6	Object cannot be sent to Offline state	25
2.5	Fault processing	26
2.5.1	Faults in the online state or request processing	26
2.5.2	Offline faults	28
2.5.3	AutoRecover attribute	29

Contents

2.5.4	Fault during offline processing	29
2.5.5	Examples of fault processing	30
2.5.6	Fault clearing	32
2.5.7	SysNode faults	33
2.5.7.1	Operator intervention	35
2.6	Switch processing	35
2.6.1	Switch request	35
2.7	Resource failure processing for applications	37
2.7.1	Online recovery with the Recover option	38
2.7.2	Offline recovery with the OfflineRecover option	38
2.7.3	Additional notes	39
2.8	Special states	39
2.8.1	Restrictions during maintenance mode	39
2.8.2	The Inconsistent state	41
2.9	RMS heartbeat operation	42
2.9.1	Operational details	44
2.9.1.1	ELM lock management	44
2.9.1.2	First node startup	45
2.9.1.3	Second node startup	45
2.9.1.4	Third and subsequent nodes	46
2.9.1.5	Node or RMS down scenario	46
2.9.1.6	Slow response scenario	47
2.9.2	Default initialization in hvenv	48
2.9.3	Manually controlling ELM in hvenv.local	48
3	Scalable controllers	49
3.1	Controller overview	49
3.2	Scalable controllers and applications	49
3.2.1	Scalable Applications	50
3.2.2	Benefits of Scalable Controllers	50
3.2.3	Attributes for Scalable Controllers	51
3.3	Online/offline processing and state transitions	52
3.3.1	How controller states depend on controlled application states	52
3.3.2	Request propagation to controlled applications	53
3.3.3	Controller warning state	53
3.3.4	Application warning state	54
3.3.5	Controller state change script	54
3.3.6	Sequenced online/standby/offline and application groups	56
3.3.7	Auto Startup on a sub-cluster	58
3.3.8	Switchover on a sub-cluster	59
3.3.9	Manual switching of child applications	59
3.3.10	Operation during maintenance mode	59
3.3.10.1	Switching applications between nodes	60
3.3.10.2	Online, offline, and fault processing restrictions	62

3.3.10.3	Recommended approach for scalable controllers	63
4	Troubleshooting	65
4.1	Overview	66
4.2	Debug and error messages	67
4.3	Log file categories	68
4.4	Managing base monitor log output	70
4.4.1	Managing base monitor log levels	70
4.4.2	Controlling base monitor output to the system log	72
4.5	Managing application log output	72
4.5.1	Controlling debug messages from standard scripts	73
4.6	Managing detector log output	74
4.6.1	Debug messages and log levels	74
4.6.2	Setting the detector log level	75
4.6.2.1	Setting the detector log level with hvutil -L	75
4.6.2.2	Setting the detector log level with PCS	76
4.6.2.3	Setting the log level in the Wizard Tools	78
4.7	Interpreting RMS log messages	79
4.7.1	switchlog message format	79
4.7.2	Application log message format	81
4.7.3	Program log message format	82
4.8	Viewing RMS log messages	83
4.8.1	Common procedures for switchlog and application log	87
4.8.2	Time filter	88
4.8.3	Keyword filters	89
4.8.3.1	Resource Name	89
4.8.3.2	Severity	90
4.8.3.3	Non-zero exit code	91
4.8.3.4	Keyword	91
4.8.4	Text search	92
4.8.5	Removing filters	92
4.9	RMS log file cleanup	93
4.9.1	hvmlogclean	93
4.9.2	hvmlogcontrol	93
4.9.3	Logging of background scripts	94
4.10	Configuration troubleshooting	94
4.10.1	PCS log and trace files	94
4.10.2	Manual script execution	95
4.10.2.1	PCS GUI method	95
4.10.2.2	PCS CUI method	97
4.10.2.3	RMS Wizard Tools method	100
4.11	RMS troubleshooting	101
4.12	Collecting information for advanced troubleshooting	105
4.12.1	Using the hvdump command (RMS)	105

Contents

5	Non-fatal error messages	107
5.1	ADC: Admin configuration	108
5.2	ADM: Admin, command, and detector queues	118
5.3	BAS: Startup and configuration errors	140
5.4	BM: Base monitor	148
5.5	CML: Command line	160
5.6	CRT: Contracts and contract jobs	161
5.7	CTL: Controllers	163
5.8	CUP: userApplication contracts	163
5.9	DET: Detectors	165
5.10	GEN: Generic detector	168
5.11	INI: init script	169
5.12	MIS: Miscellaneous	170
5.13	QUE: Message queues	170
5.14	SCR: Scripts	170
5.15	SWT: Switch requests (hvswitch command)	172
5.16	SYS: SysNode objects	173
5.17	UAP: userApplication objects	177
5.18	US: us files	182
5.19	WLT: Wait list	183
5.20	WRP: Wrappers	184
6	Fatal error messages	191
6.1	ADC: Admin configuration	192
6.2	ADM: Admin, command, and detector queues	193
6.3	BM: Base monitor	193
6.4	CML: Command line	196
6.5	CMM: Communication	196
6.6	CRT: Contracts and contract jobs	197
6.7	DET: Detectors	197
6.8	INI: init script	198
6.9	MIS: Miscellaneous	201
6.10	QUE: Message queues	201
6.11	SCR: Scripts	202
6.12	SYS: SysNode objects	204
6.13	UAP: userApplication objects	205
6.14	US: us files	205
6.15	WRP: Wrappers	206
7	Console error messages	209
7.1	Console messages in alphabetical order	209
8	Warning messages	233
8.1	ADC: Admin configuration	234

8.2	ADM: Admin, command, and detector queues	235
8.3	BAS: Startup and configuration errors	238
8.4	BM: Base monitor	239
8.5	CTL: Controllers	243
8.6	CUP: userApplication contracts	244
8.7	DET: Detectors	251
8.8	SCR: Scripts	251
8.9	SWT: Switch requests (hvswitch command)	252
8.10	SYS: SysNode objects	259
8.11	UAP: userApplication objects	261
8.12	US: us files	263
8.13	WLT: Wait list	265
8.14	WRP: Wrappers	266
9	Notice messages	267
9.1	ADC: Admin configuration	267
9.2	BM: Base monitor	268
9.3	SWT: Switch requests (hvswitch command)	268
9.4	SYS: SysNode objects	269
9.5	US: us files	269
9.6	WRP: Wrappers	271
10	Appendix—Operating system error numbers	273
10.1	Solaris error numbers	273
10.2	Linux error numbers	279
11	Appendix—States	285
11.1	Basic states	285
11.2	State details	287
12	Appendix—Object types	289
13	Appendix—Attributes	291
13.1	Attributes available to the user	291
13.2	Attributes managed by configuration wizards	299
14	Appendix—Environment variables	303
14.1	Setting environment variables	303
14.2	Global environment variables	304
14.3	Local environment variables	309
14.4	Script execution environment variables	313
15	Appendix—RMS command line interface	315
15.1	Available RMS CLI commands	315

Contents

16	Appendix—List of manual pages	319
16.1	CCBR	319
16.2	CF	319
16.3	CIP	320
16.4	PAS	320
16.5	Resource Database	320
16.6	RMS	322
16.7	SF	323
16.8	Monitoring Agent	324
16.9	SIS	325
16.10	Web-Based Admin View	325
16.11	RMS Wizards	326
16.12	Miscellaneous utilities	326
Glossary		327
Abbreviations		343
Figures		347
Tables		349
Index		351

1 Preface

Reliant[®] Monitor Services (RMS) is a software monitor designed to guarantee the high availability of applications in a cluster of nodes. This manual describes how to troubleshoot RMS configuration and operation.

This manual is aimed at RMS experts who create and tune RMS configurations. It assumes an understanding of all the basic information in the RMS configuration guides. The reader should be familiar with PCS or the RMS Wizard Tools, the Cluster Admin GUI, the RMS CLI commands and utilities, and the following system functions and components:

- PRIMECLUSTER family of products
- Linux[®] or Solaris[™] operating system
- Data center components such as volume managers and storage area networks.

1.1 About this manual

This manual is structured as follows:

- The chapter “Advanced RMS concepts” on page 9 provides background details about RMS operation including state detection and transition processing.
- The chapter “Scalable controllers” on page 49 describes operational details of scalable controllers and scalable applications.
- The chapter “Troubleshooting” on page 65 describes how to troubleshoot RMS using graphical user interface (GUI) and command line interface (CLI) tools.
- The chapter “Non-fatal error messages” on page 107 lists all RMS error messages written to the log file along with their causes and resolutions.
- The chapter “Fatal error messages” on page 191 lists all fatal RMS error messages written to the log file along with their causes and resolutions.
- The chapter “Console error messages” on page 209 lists all RMS error messages written to the console along with their causes and resolutions.
- The chapter “Warning messages” on page 233 lists all RMS warning messages written to the log file along with their causes and resolutions.

- The chapter “Notice messages” on page 267 lists selected RMS notice messages written to the log file along with their causes and resolutions.
- The chapter “Appendix—Operating system error numbers” on page 273 lists operating system error numbers for Solaris and Linux.
- The chapter “Appendix—States” on page 285 lists the object states that are supported by RMS.
- The chapter “Appendix—Object types” on page 289 lists the object types that are supplied with RMS.
- The chapter “Appendix—Attributes” on page 291 lists the attributes that are supported by RMS object types.
- The chapter “Appendix—Environment variables” on page 303 describes the RMS environment variables.
- The chapter “Appendix—RMS command line interface” on page 315 lists the RMS administrative CLI commands.
- The chapter “Appendix—List of manual pages” on page 319 lists the manual pages for PRIMECLUSTER.

1.2 PRIMECLUSTER documentation list

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

- Release notices for all products—These documentation files are included as HTML files on the PRIMECLUSTER CD. Release notices provide late-breaking information about installation, configuration, and operation. Read this information first.
- *Concepts Guide (Solaris, Linux)*—Provides conceptual details on the PRIMECLUSTER family of products.
- *PRIMECLUSTER Installation and Administration Guide (Solaris)*—Provides instructions for installing and upgrading PRIMECLUSTER products.
- *PRIMECLUSTER Installation and Administration Guide (Linux)*—Provides instructions for installing and upgrading PRIMECLUSTER products.
- *Web-Based Admin View (Solaris) Operation Guide*—Provides information on using the Web-Based Admin View management GUI.

- *Web-Based Admin View (Linux) Operation Guide*—Provides information on using the Web-Based Admin View management GUI.
- *Cluster Foundation (CF) (Solaris) Configuration and Administration Guide*—Provides instructions for configuring and administering the PRIMECLUSTER Cluster Foundation.
- *Cluster Foundation (CF) Configuration and Administration Guide (Linux)*—Provides instructions for configuring and administering the PRIMECLUSTER Cluster Foundation.
- *Reliant Monitor Services (RMS) with Wizard Tools (Linux, Solaris) Configuration and Administration Guide*—Provides instructions for configuring and administering PRIMECLUSTER Reliant Monitor Services using the Wizard Tools interface.
- *Reliant Monitor Services (RMS) with PCS (Linux, Solaris) Configuration and Administration Guide*—Provides instructions for configuring and administering PRIMECLUSTER Reliant Monitor Services using the PCS (PRIMECLUSTER Configuration Services) interface.
- *Reliant Monitor Services (RMS) (Linux, Solaris) Reference Guide*—Describes operational principles and diagnostic procedures for the RMS high availability manager, 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) (Linux, Solaris) Configuration and Administration Guide*—Provides information on configuring and administering Scalable Internet Services (SIS).
- *Global Disk Services (Solaris, Linux) Configuration and Administration Guide*—Provides information on configuring and administering Global Disk Services (GDS).
- *Global File Services (Solaris, Linux) Configuration and Administration Guide*—Provides information on configuring and administering Global File Services (GFS).
- *Global Link Services (Solaris, Linux) 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, Linux) Configuration and Administration Guide: Multipath Function*—Provides information on configuring and administering the multipath function for Global Link Services (GLS).

- *Data Management Tools (Solaris) Configuration and Administration Guide*—Provides reference information on the Volume Manager (RCVM) and File Share (RCFS) products. (Not available in all markets)
- *SNMP Reference Manual (Solaris, Linux)*—Provides reference information on the Simple Network Management Protocol (SNMP) product.
- *RMS Wizards documentation package*—Available on the PRIMECLUSTER CD. These documents deal with Wizard Tools topics such as the configuration of file systems and IP addresses. They also describe the various types of available RMS wizards.

1.3 Conventions

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

1.3.1 Notation

This manual uses the following notational conventions.

1.3.1.1 Prompts

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

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

1.3.1.2 Manual page section numbers

References to operating system commands may sometimes be followed by their manual page section numbers in parentheses, *e.g.*, `cp(1)`.

1.3.1.3 The keyboard

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

1.3.1.4 Typefaces

The following typefaces highlight specific elements in this manual.

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

Typeface conventions are shown in the following examples.

1.3.1.5 Example 1

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

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

1.3.1.6 Example 2

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

```
$ cat file
```

1.3.2 Command line syntax

The command line syntax observes the following conventions.

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

1.4 Important notes and cautions

Material of particular interest is preceded by one of the following symbols:



Contains important information about the subject at hand.

**Caution**

Indicates a situation that can cause harm to data.

1.5 Abbreviations

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

1.6 Editing record

Additions and changes	Section	Manual code
Added a note about eliminating RMS in Maintenance mode.	2.8.1	J2UZ-5294-02ENZ0(01)
Changed an attribute value.	13.1	
Changed the explanation about Unexpected reports during Online processing.	2.3.4	J2UZ-5294-02ENZ0(02)
Changed the explanation about Unexpected reports during Offline processing.	2.4.3	
Changed the conditions for error occurrence during Online processing.	2.3.5	
Changed the conditions for error occurrence during Offline processing.	2.4.4	
Changed the possible values of HV_CONNECT_TIMEOUT.	14.3	
		J2UZ-5294-02ENZ0(03)

Additions and changes	Section	Manual code
Changed the action of (BM, 29).	5.4	J2UZ-5294-02ENZ0(04)
Deleted the attribute "OnlineTimeout."	13.1	
Deleted the description of Master process.	5.12 6.9	
Deleted the following messages:	8.2	
- (ADM, 117)	9.1	
- (ADC, 64)	9.2	
- (ADM, 36)	9.3	
- (SWT, 81)	9.5	
- (US, 38)		
- (US, 39)		
- (US, 54)		

2 Advanced RMS concepts

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

Chapter contents:

- “Internal organization” on page 9
- “Initializing” on page 10
- “Online processing” on page 15
- “Offline processing” on page 22
- “Fault processing” on page 26
- “Switch processing” on page 35
- “Resource failure processing for applications” on page 37
- “Special states” on page 39
- “RMS heartbeat operation” on page 42

2.1 Internal organization

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

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

2.1.1 Application and resource description

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

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

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

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

2.1.2 Messages

RMS objects exchange messages for the following purposes:

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

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

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

2.2 Initializing

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

The following is necessary information for identifying the state:

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

Two conclusions can be drawn from the above:

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



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

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

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

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

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

Example configuration

The examples of RMS processing in the following sections are based on an application `app` configured to run on `shasta1RMS` and `shasta2RMS` as follows:

- For each node, `shasta1RMS` and `shasta2RMS`, there is one `SysNode` object bearing the name of the node.

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

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

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

A diagram of the object hierarchy is shown in Figure 1.

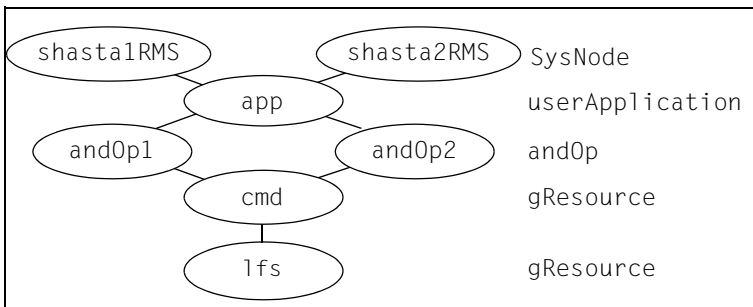


Figure 1: Object hierarchy for initializing examples

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

The actual RMS graph for this configuration as produced by PCS is shown in Figure 2.

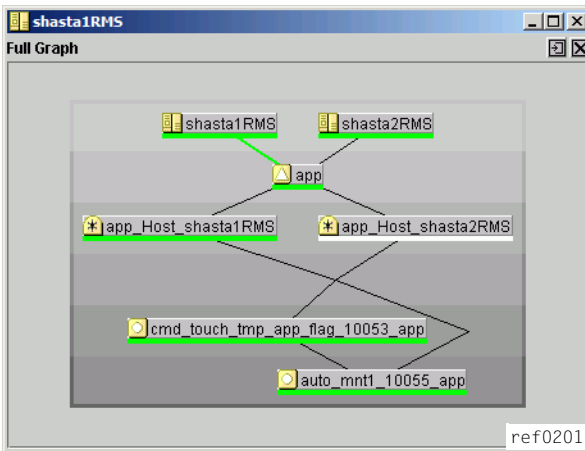


Figure 2: System graph for initializing examples—PCS

The title bar contains the name of the node on which the graph was drawn, which is `shasta1RMS`. The line connecting `shasta1RMS` to `app` is green, indicating the node where the application is online.

The corresponding output from `hvdisp -a` is shown in Figure 3.

```
shasta1:~ # hvdisp -a

Local System: shasta1RMS
Configuration: /opt/SMAW/SMAWpcs/Config/ref_example/runtime/ref_example.us

Resource          Type      HostName          State      StateDetails
-----
shasta2RMS        SysNode
shasta1RMS        SysNode
app               userApp
app_Host_shasta2RMS andOp    shasta2RMS
app_Host_shasta1RMS andOp    shasta1RMS
cmd_touch_tmp_app_flag_10053_app gRes
auto_mnt1_10055_app gRes
```

Figure 3: hvdisp output for initializing examples—PCS

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

Note that some of the actual object names generated by PCS are more complex than the simplified, generic names shown in the abstract graph of Figure 1. Also, the graph contains additional dependencies that are automatically inserted to ensure proper operation. Selecting other items from the Cluster Admin *Preferences* menu will display further detail and reveal additional objects and dependencies.

If the same example is configured in the Wizard Tools, the generated resource names will have a slightly different format. However, the structure of the graph will be identical.

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

Example 1

The following process for the configuration illustrated in Figure 1 is applicable for a monitor running on `shasta1RMS`:

1. RMS starts.
2. The base monitor determines the state of the `SysNode` objects.
3. The detectors of the `cmd` and `lfs` resources report their respective states as `Offline`.
4. Since it is a leaf object, `lfs` changes immediately to `Offline` and reports this state change to its parent.
5. After receiving the detector report and the report of its child, `cmd` possesses the necessary information for determining its own state. It then goes offline and notifies this change of state to its parent `andOp1`.



`andOp2` is a remote object which is ignored by the base monitor on `shasta1RMS`.

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

2.3 Online processing

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

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

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

These situations are discussed in detail in later sections.

2.3.1 Online request

Generating the online request is referred to as **switching** the `userApplication`; that is, switching the `userApplication` online or switching the `userApplication` to another cluster node (refer also to the section “Switch processing” on page 35).

The following actions can generate an online request:

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

2.3.1.1 Manual methods

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

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

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

Manual request using the GUI

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

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

Manual request using the CLI

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

2.3.1.2 Automatic methods

All automatic methods can only invoke a priority switch.

Automatic request at RMS startup

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

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

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

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

Automatic request when a fault occurs

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

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

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

2.3.2 PreCheckScript

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

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

Resulting state

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

AutoSwitchOver

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

2.3.3 Online processing in a logical graph of a userApplication

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

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

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

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

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

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

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

i How a `cmdLine` script influences the state of the actual application depends on the application itself. RMS has no direct control over any user application. For a more complete discussion, see the section “Relationship of RMS configurations to the real world” in the introductory chapter of any RMS configuration and administration guide.

Example 2

The scenario for this example is as follows:

- `AutoStartup` attribute is set to 1.
- None of the resource objects have `PreOnlineScript` definitions.
- All objects are in the `Offline` state at startup time.

Online processing is as follows:

1. RMS starts.
2. `userApplication` object `app` on node `shasta1RMS` generates a pre-online request because the `AutoStartup` attribute is set to 1.
3. This request is passed through to the `lfs` leaf object. As no `PreOnlineScript` has been configured for any of the objects in this example, `lfs` forwards a message to `app` indicating that pre-online processing has completed successfully.
4. When the pre-online success message arrives, `app` generates the online request, which is also passed through to the `lfs` leaf object.
5. The `lfs` object executes the online script and brings the disk online.
6. As soon as the detector of `lfs` reports `Online`, successful completion of online processing is notified upwards to the `cmd` object. (If the object had a post-online script, this would have been executed before the success message was forwarded.)
7. The `cmd` object starts its online script.
8. As soon as the `cmd` detector reports a success completion, the success message is forwarded to `andOp1`.

9. The `andOp1` object is a object without a detector; it does not have an online script in this example. As soon as its local child reports the `Online` state, it forwards the success message to its parent object `app`.
10. Upon receipt of the success message at `app`, RMS executes the online script and the application starts. Because `app` does not have a detector and also because no post-online script is configured, `app` changes immediately to the `Online` state after the online script has completed successfully.

2.3.4 Unexpected reports during online processing

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

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

For cases where the object's `online` processing fails, see the section “Fault situations during online processing” on page 20.

2.3.5 Fault situations during online processing

If an error situation occurs during `online` processing, the affected object commences `fault` processing and notifies its parent of the error (see also the section “Fault processing” on page 26). The following can cause faults during `online` processing:

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

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

2.3.6 Initialization when an application is already online

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

Request while online

If a `userApplication` receives an online request when it is already online, it is forwarded to the other objects as usual. The only difference from the description in the section “Online processing” on page 15 is that any objects that are already online forward the request or the responses without executing their scripts and without changing to the `Wait` state. In particular, the `PreCheckScript` is not run.

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

No request while online

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

Guarding against data loss when the application is already online

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



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

**Caution**

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

2.4 Offline processing

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

2.4.1 Offline request

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

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

In normal operating mode, only the RMS command interface can generate an offline request. In the case of a fault, the `userApplication` generates its own offline request (such as if one or more necessary resources fails); this prevents an application that is no longer operating correctly from continuing to operate in an uncontrolled manner (see also the section “Fault processing” on page 26). This offline request is also a primary precondition for any subsequent switchover.



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

2.4.2 Offline processing in a logical graph of a userApplication

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

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

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

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

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

Example 3

The following further explains the offline process:

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

2.4.3 Unexpected reports during offline processing

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

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

For cases where the object's `offline` processing fails, see the section “Fault situations during offline processing” on page 25.

2.4.4 Fault situations during offline processing

If an error situation occurs during `offline` processing, the affected object commences `fault` processing and notifies its parent of the error (see also the section “Fault processing” on page 26). The following can cause faults during `offline` processing:

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

2.4.5 Object is already in Offline state

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

2.4.6 Object cannot be sent to Offline state

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

During `offline` processing, an object whose `LieOffline` attribute is set reacts in the same way as any other object when its pre-offline, `offline`, and post-offline scripts are run. The reaction of the object with respect to its parent is also the same as if the object had been successfully taken `offline`; that is, it “lies.” A

object with `LieOffline` set does not wait for an offline report of the detector after the offline script has executed; instead, it automatically executes the post-offline script. An unexpected online report of the detector (which arrives after the offline script has executed) is not a fault condition in this case.

2.5 Fault processing

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

2.5.1 Faults in the online state or request processing

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

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

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

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


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

AutoSwitchOver includes ResourceFailure

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


1. The `userApplication` attempts to initiate the switchover procedure. For this purpose, the application on the local node must be set to a defined `Offline` state. The procedure is the same as that described under offline processing.
2. When offline processing is successfully completed, an online request is sent to the corresponding `userApplication` of a remote node (see the section “Switch processing” on page 35). However, the `userApplication` is now in the `Faulted` state—unlike the situation with a normal offline request. This prevents the possibility of an application returning to the node in the event of another switchover.

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

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

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

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

 A double fault causes the node to be eliminated if the application's `Halt` attribute is set and the application can be switched to another node.

AutoSwitchOver does not include ResourceFailure and PreserveState=1

In this case, the process is as follows:

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

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

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

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

Fault during pending switch request

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



For more information about the `AutoSwitchOver` and `PreserveState` attributes, see the chapter “Appendix—Attributes” on page 291.

2.5.2 Offline faults

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

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

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

2.5.3 AutoRecover attribute

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

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

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

2.5.4 Fault during offline processing

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

2.5.5 Examples of fault processing

The following are examples of fault processing.

Example 4

The scenario for this example is as follows:

- The application `app` has its `AutoSwitchOver` attribute set to `ResourceFailure` and is online on node `shasta1RMS`.
- There is no request.
- The `lfs` object does not have its `AutoRecover` attribute set.
- An error by the system administrator unmounts the `lfs` file system.

Fault processing is as follows:

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



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

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

A more complex offline script for `lfs` could check whether the object is still mounted and terminate with an exit status of 0. In this case, RMS could successfully complete offline processing after the fault and switch over to `shasta2RMS`; all local objects on `shasta1RMS` would then be offline following successful online processing, and only `app` would remain in the `Faulted` state.

Example 5

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

Fault processing is as follows:

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

Example 6

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

Fault processing is as follows:

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



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

2.5.6 Fault clearing

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

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

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

```
hvutil -c userApplication
```

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

If the `userApplication` object is initially online, invoking `'hvutil -c'` may not affect every object in the tree. If the graph has an `orOp` that was also initially online, the online processing will treat that `orOp` as a leaf object (the end of its branch). Objects below the `orOp` may continue to be in the faulted state as long as at least one of the children of the `orOp` is online. To initiate online or offline processing for the entire tree, use `'hvswitch -f'` or `'hvutil -f'` as described below.

2. The following command makes a forced online request:

```
hvsswitch -f userApplication target_node
```

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

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

```
hvutil -f userApplication
```

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

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

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

2.5.7 SysNode faults

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

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

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

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

Example 7

The scenario for this example is as follows:

- RMS is running on a cluster consisting of nodes `shasta1` and `shasta2`, which are represented by the `SysNode` objects `shasta1RMS` and `shasta2RMS`, respectively.
- `app` is online on `shasta1RMS` and its `AutoSwitchOver` attribute includes the `HostFailure` setting.
- A system fault on `shasta1` generates a panic message.

The reaction of RMS is as follows:

1. CF determines that a node failure has occurred and generates a `LEFTCLUSTER` event.
2. RMS puts the `SysNode` in a `Wait` state. RMS receives the `LEFTCLUSTER` event and sends a kill request to SF.
3. After SF successfully kills the node, a `DOWN` event is sent.
4. RMS receives the `DOWN` event and marks the `SysNode` as `Failed`.
5. The `shasta1RMS` object executes its fault script (assuming that such a script has been configured).
6. The `shasta1RMS` object notifies the `userApplication` objects that `shasta1RMS` has failed. Since `app` was online on `shasta1RMS` when `shasta1RMS` failed, and since its `AutoSwitchOver` attribute includes the `HostFailure` setting, the object `app` on `shasta2RMS` starts online processing.

2.5.7.1 Operator intervention

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

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

For more information about the `ScriptTimeout` attribute, see the chapter “Appendix—Attributes” on page 291.

2.6 Switch processing

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

2.6.1 Switch request

Switch requests are divided as follows:

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

The types of switches are divided as follows:

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



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

Example 8

The scenario for this example is as follows:

- app is online on shasta1RMS.
- The system administrator sends a directed switch request on shasta2RMS with the aim of switching app to shasta2RMS.

Switch processing is as follows:

1. RMS forwards the switch request to shasta1RMS because shasta1RMS is the online node of the userApplication object.
2. app on shasta1RMS notifies the corresponding nodes in the cluster (in this case, app on shasta2RMS) that switchover processing is active. This means that competing activities are blocked.
3. app on shasta1RMS sends a request to app on shasta2RMS to establish whether any faults are known in the local graph on shasta2RMS. In this example, there are no known faults in the local graph on shasta2RMS.
4. app on shasta1RMS commences offline processing.
5. As soon as RMS has successfully deconfigured app on shasta1RMS, app on shasta1RMS notifies all corresponding nodes in the cluster that the application will now be running on shasta2RMS. Blocking of competing requests is simultaneously cancelled.
6. app on shasta1RMS terminates its activity by sending an explicit online request to app on shasta2RMS.
7. app on shasta2RMS commences online processing.

2.7 Resource failure processing for applications



Caution

Resource failure processing for applications is not supported.

The `AutoSwitchOver` attribute of a `userApplication` controls autorecovery processing when resource failures in the graph occur. The possibilities depend on the options included in the `AutoSwitchOver` settings.

Local recovery is governed by the `Recover` and `OfflineRecover` options:

- re-invoke local online processing: `Recover` is specified
- invoke local offline processing followed by local online processing: `OfflineRecover` is specified

Switchover to another node is governed by the `ResourceFailure` option:

- failover to another node: `ResourceFailure` is specified
- take application offline: `ResourceFailure` is not specified and the `PreserveState` attribute is 0
- do nothing: `ResourceFailure` is not specified and the `PreserveState` attribute is 1

The `ResourceFailure`, `Recover`, and `OfflineRecover` options may be combined with other `AutoSwitchOver` options using the symbolic *OR* (vertical bar) syntax, as in the following example.

```
AutoSwitchOver=HostFailure|ResourceFailure|Recover
```

The `Recover` and `OfflineRecover` options are mutually exclusive and must not be specified together. However, either one can be used with the `ResourceFailure` option: if a local autorecovery attempt fails, then failover to another node is attempted according to the `ResourceFailure` setting.

Details of the `Recover` and `OfflineRecover` processing are presented below.

2.7.1 Online recovery with the Recover option

As soon as a resource failure in the graph propagates to the `userApplication`, online processing is (re-)started. The behavior is the same as if the administrator had issued a “forced online request” via `hvswitch -f`. All the usual rules for online processing apply. In particular, all objects which are already online propagate the request, but they do not invoke their online scripts.

- If the online processing is successful, the `userApplication` finally transits back into the online state. No further failover reactions are invoked.
- If the online processing fails, normal fault processing takes place according to the other attribute settings of the `userApplication`.

2.7.2 Offline recovery with the OfflineRecover option

As soon as a resource failure in the graph propagates to the `userApplication`, offline processing is (re-)started. The behavior is the same as if the administrator had issued a “clear request” via `hvutil -c`. All the usual rules for offline processing apply.

- If the offline processing is successful it is immediately followed by online processing. Again, the behavior is the same as if the administrator had issued a “unforced online request” via `hvswitch`. All the usual rules for online processing apply.
- If the online processing fails, normal fault processing takes place according to the other attribute settings of the `userApplication`.
- However, if the offline processing had already been attempted and failed, this is considered a **double fault**: this basically means that offline processing was unable to bring the graph into a settled offline state after a previous resource failure.

In this case, the normal rules for the double fault apply. In particular, no further online processing is invoked, and no failover takes place. Intervention by the system administrator is required. If the `Halt` attribute is set, the node is eliminated.

2.7.3 Additional notes

The rules above also apply if the `userApplication` is in the standby state. The only difference is that standby processing is invoked instead of online processing.

Like autorecovery for single resource objects, autorecovery for a `userApplication` is invoked only if the `userApplication` is in a settled online or standby state. In particular, it is not invoked if an ongoing online, standby, offline, or fault processing results in a failure.

All autorecovery attempts of the `userApplication`, including their success or failure, are logged in the `switchlog`.

2.8 Special states

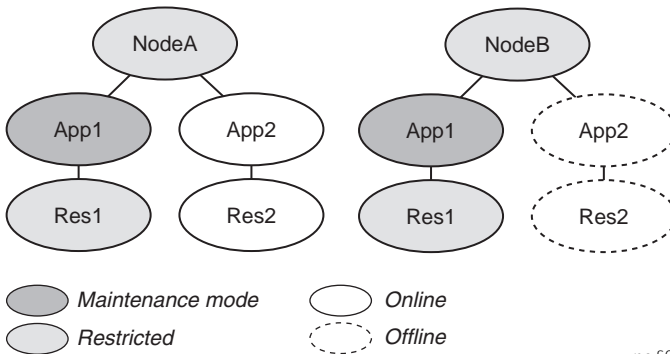
2.8.1 Restrictions during maintenance mode

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

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

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

To illustrate how the restrictions apply, consider the situation in which two applications (`App1` and `App2`) each have a single dependent resource (`Res1` and `Res2`, respectively) and are configured to run on either of two nodes (`NodeA` and `NodeB`). Figure 4 shows the states of these objects if `App1` is put into maintenance mode.



ref0202

Figure 4: Example of maintenance mode restrictions

This simple example illustrates the following features:

- Maintenance mode for an application applies on every node where the application could be brought online, *i.e.*, on every node in its *PriorityList* attribute. Therefore, App1 is in the Maintenance state on every node where it could run, regardless of its previous state on any of those nodes.

Note that App1 also has an **intended state** on each of its nodes. This is the target state for the application when it comes out of maintenance mode. The intended state is available in the application's *StateDetails* parameter in the GUI and in the CLI `hvdisp` output.)

- Since Res1 is a child of App1, it is restricted everywhere.
- Since App1 appears in the graph of both NodeA and NodeB, they are also restricted. (You can initiate RMS shutdown of either node, but you will be prompted to let RMS take App1 out of maintenance mode first.)
- App1 does not appear in the graph of App2. Therefore, normal processing of App2 and its resource is allowed, including switching App2 offline, online, or to a different node.
- Do not stop RMS or the system while the application is in the maintenance mode. You must exit the maintenance mode of all applications before stopping them.

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

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

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

2.8.2 The Inconsistent state

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

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

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

Note that while the userApplication is displayed or reported as Inconsistent, this is not a true state in the RMS state machine: the true state is either Offline or Faulted. For most operations, the behavior of an Inconsistent userApplication is determined by the underlying true state. For instance, if the true state is Offline, and an Offline request is issued, no Offline script will be fired (see the section "Object is already in Offline state" on page 25).

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

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



Caution

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

2.9 RMS heartbeat operation

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

RMS transmits a UDP **heartbeat** signal at regular intervals. If the elapsed time since the last heartbeat from a node exceeds an adjustable connection timeout, RMS assumes the node has lost connectivity (see “`HV_CONNECT_TIMEOUT`” on page 309). RMS then begins a recovery period for the node. If the node heartbeat is detected during the recovery period, RMS assumes the node is functional and returns it to normal status. However, if RMS receives no heartbeats from the node before the recovery period expires, it assumes the node is down, even if other communication with the node is possible.

Once RMS marks a node as down, it takes a series of steps to ensure application and cluster integrity. First, it is necessary to ensure that the node is truly shut down. Otherwise, the node and its applications could unexpectedly recover later, causing conflicts and data corruption. To avoid these problems, RMS directs the Shutdown Facility (described later) to eliminate the node. This is often done by rebooting the node or turning off its power, but the exact action depends on which shutdown agents have been configured for the node. Only after the node has been eliminated is it safe for RMS to restart the node's applications elsewhere in the cluster. The process of automatically switching applications from a failed node to a healthy node is called application **failover**.

Application switchover impacts cluster performance, so it is important to choose a recovery timeout that avoids false detection of node outages. The optimum UDP recovery time depends on the conditions in the cluster. A short recovery period is the best choice to deal with failures of nodes or base monitors. However, a long recovery period allows time for overloaded nodes to respond, which avoids unnecessary shutdowns. If the UDP method is used by itself, these opposing requirements make it difficult to tune the recovery time in large or busy clusters. When CF is not present, as is the case in a SAP cluster,

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

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

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

Experts can manually disable ELM for rolling upgrade or debugging operations (see "HV_USE_ELM" on page 307). In this case, when RMS starts up, the expert must also manually adjust the UDP heartbeat recovery timeout to a smaller value with `hvcn -h <timeout>`. This is necessary to efficiently detect remote base monitor and node outages in the absence of ELM.

2.9.1 Operational details

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

2.9.1.1 ELM lock management

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

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

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

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

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

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

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

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

2.9.1.2 First node startup

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

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

2.9.1.3 Second node startup

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

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

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

2.9.1.4 Third and subsequent nodes

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

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

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

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

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

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

2.9.1.5 Node or RMS down scenario

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

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

1. A's pending request to convert B's lock to the CR state wakes up. A now holds that lock in the CR state.
2. A immediately converts the lock to the NULL state and marks B as being offline.
3. A continues to mark B as offline until it receives a heartbeat from B. At that point, A restarts the lock cycle by requesting to convert B's lock to the CR state.

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

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

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

2.9.1.6 Slow response scenario

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

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

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

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

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

2.9.2 Default initialization in hvenv

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

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

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



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

2.9.3 Manually controlling ELM in hvenv.local

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

- rolling upgrades
- testing or debugging

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

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

See “`HV_USE_ELM`” on page 307.

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

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



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

3 Scalable controllers

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

- “Controller overview” on page 49
- “Scalable controllers and applications” on page 49
- “Online/offline processing and state transitions” on page 52

3.1 Controller overview

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

3.2 Scalable controllers and applications

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

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

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

3.2.1 Scalable Applications

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

3.2.2 Benefits of Scalable Controllers

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

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

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

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

3.2.3 Attributes for Scalable Controllers

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

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

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

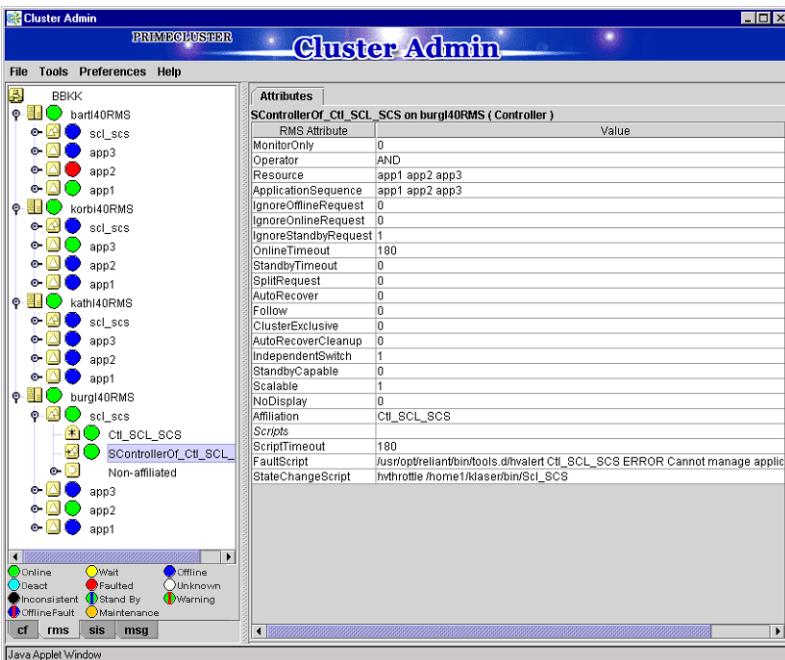


Figure 5: Scalable controller attributes

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

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

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

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

3.3 Online/offline processing and state transitions

3.3.1 How controller states depend on controlled application states

Table 1 summarizes scalable controller states according to the states of its controlled applications:

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

Table 1: Dependence of scalable controller states on child applications

Note: Although two or more different scalable applications can be online at the same time on the same or on different hosts, each scalable application can still be online only on a single host. For example, when creating an RMS configuration for Oracle RAC, the Oracle daemon processes that must be online on multiple hosts should be represented not by a single RMS application, but by a number of separate, controlled RMS applications—one child application per daemon. The parent scalable controller for Oracle RAC becomes a single point of control, administration, and display for all these child applications.

3.3.2 Request propagation to controlled applications

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

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

3.3.3 Controller warning state

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

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

3.3.4 Application warning state

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

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

3.3.5 Controller state change script

The controller's `StateChangeScript` attribute specifies a script to be executed for a scalable controller whenever a controlled child application transitions into an online, offline, faulted, or standby state. The script is executed on every host where the controlling parent application can run, even if it is done on behalf of a request originated from the controller itself. The state change script will also be executed if a host where a child application is running changes its state to offline or faulted. This script has no impact on state transitions—its abnormal exit code is merely recorded in the `switchlog`.

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

Script execution environment variables

The state change script's environment contains a set of variables that can be used for decision processing at runtime. Their names and purposes are summarized below. For a complete description of their content and format, see the section "Global environment variables" on page 304, the section "Script execution environment variables" on page 313.

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

- **HV_APPLICATION_STATE_CHANGE_INFO**

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

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

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

- **HV_HOST_STATE_CHANGE_INFO**

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

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

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

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

- **HV_APPLICATION**

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

- **HV_AUTORECOVER**

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

- **HV_FORCED_REQUEST**

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

- **HV_LAST_DET_REPORT**

Last detector report for the current object.

- **HV_OFFLINE_REASON**

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

- **HV_NODENAME**

Name of current object.

- **HV_SCRIPT_TYPE**

Script type.

- **NODE_SCRIPTS_TIME_OUT**

Timeout value for the current object and script type.

Script execution sequence

The state change script is executed immediately upon an event. As a result, it can run in parallel with other RMS scripts, including state change scripts for other objects or even for the current object. Since several events may be delivered nearly at the same time, the user's script is responsible to work in proper sequence. The writer of the script must assure proper access to shared resources from this and other scripts by using locks or other OS means.

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

3.3.6 Sequenced online/standby/offline and application groups

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

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

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

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

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

The controller would issue online requests as follow:

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

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

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

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

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

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

3.3.7 Auto Startup on a sub-cluster

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

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

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

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

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

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

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

3.3.8 Switchover on a sub-cluster

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

3.3.9 Manual switching of child applications

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

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

3.3.10 Operation during maintenance mode

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

- The configuration consists of applications App1, App2, and App3 on nodes `shasta1` and `shasta2`.
- App2 controls App1 with a follow mode controller.
- App3 controls App2 with a scalable mode controller.

Figure 6 shows the RMS configuration tree and the corresponding graph for this scenario.

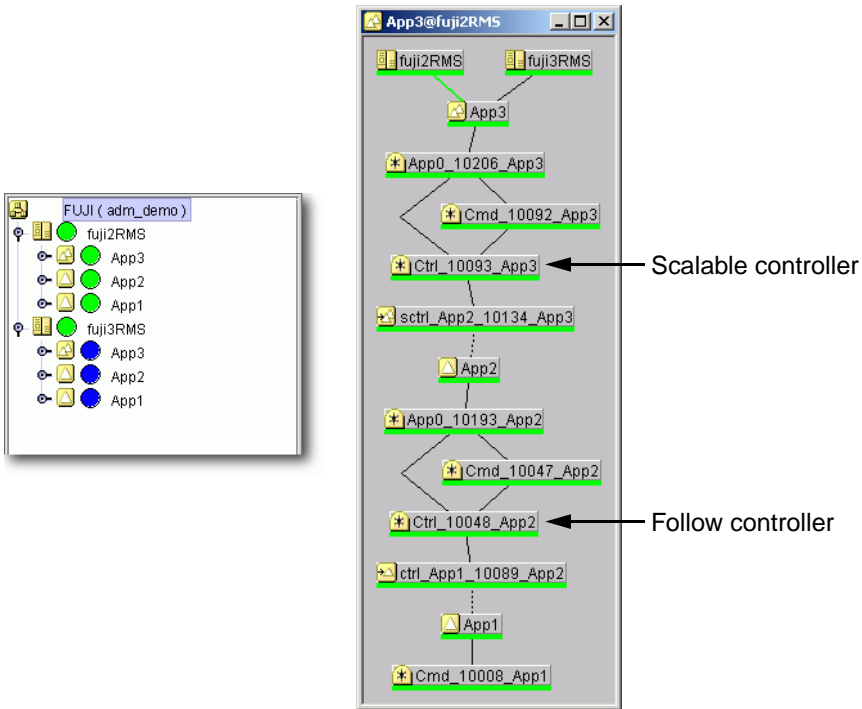


Figure 6: Maintenance mode controller example

3.3.10.1 Switching applications between nodes

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

If a scalable-mode child application is put into maintenance mode, the parent can still be switched between nodes with `hvs w i t c h`, either manually or automatically (Figure 7). Note that putting `App2` into maintenance mode automatically puts its follow-mode child application, `App1`, into maintenance mode as well.

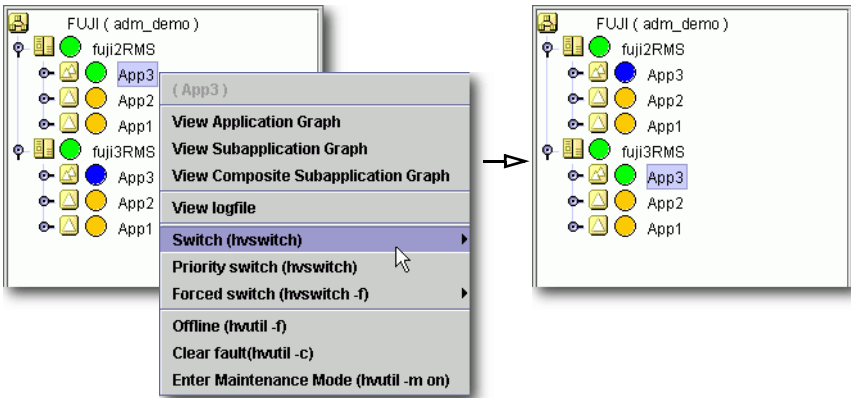


Figure 7: Switching scalable parent with child in maintenance mode

Likewise, if a scalable-mode parent application is put into maintenance mode, the child can still be switched between nodes with `hvswitch` (Figure 8). However, in actual practice, this would only occur via manual commands because the parent issues no processing requests while in maintenance mode. Note that switching `App2` to a different node automatically switches its follow-mode child application, `App1`, as well.

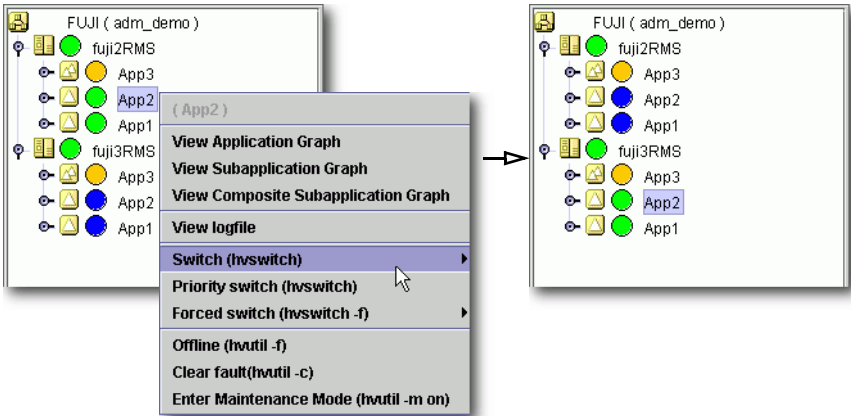


Figure 8: Switching scalable child with parent in maintenance mode

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

3.3.10.2 Online, offline, and fault processing restrictions

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

Consider what happens if you attempt to take App3 offline with `hvuтил -f` while App2 is in maintenance mode (Figure 9).

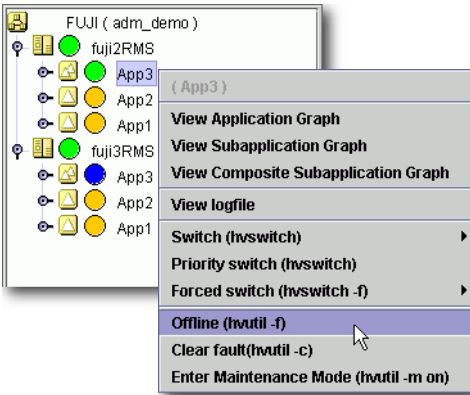


Figure 9: Attempting to take a parent offline with the child in maintenance mode

Unlike the `hvs witch` operations described previously, this `'hvuтил -f'` operation requires a response from its child application before it can proceed. Since maintenance mode prevents App2 from responding to normal processing requests from its parent, the offline request will eventually time out. The same situation occurs when fault processing via `'hvuтил -c'` is attempted.

Attempting to take a child offline while the parent is in maintenance mode (Figure 10) also fails, but there is no long delay as in the previous case: RMS immediately generates an error because only the parent can initiate offline processing.

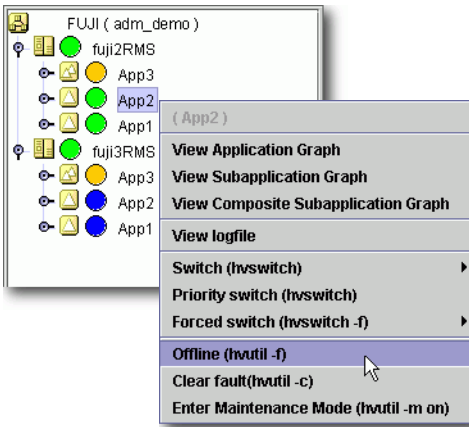


Figure 10: Attempting to take a child offline with the parent in maintenance mode

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

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

3.3.10.3 Recommended approach for scalable controllers

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

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

4 Troubleshooting

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

Chapter contents:

- “Overview” on page 66 summarizes the troubleshooting process.
- “Debug and error messages” on page 67 describes RMS debug and error messages.
- “Log file categories” on page 68 classifies the RMS log files.
- “Managing base monitor log output” on page 70 describes how to manage logs produced by the RMS base monitor and its components.
- “Managing application log output” on page 72 describes how to manage logs produced by standard PCS or Wizard Tools application and resource components.
- “Managing detector log output” on page 74 describes how to manage logs produced by standard PCS or Wizard Tools resource detectors.
- “Interpreting RMS log messages” on page 79 describes the formats of messages produced by the components of the RMS base monitor, and the applications and resources configured by PCS or the Wizard Tools.
- “Viewing RMS log messages” on page 83 describes the log viewer facility available in the Cluster Admin GUI.
- “RMS log file cleanup” on page 93 describes how to control the volume of RMS log files either automatically or manually.
- “Configuration troubleshooting” on page 94 describes troubleshooting resources available in PCS and the Wizard Tools.
- “RMS troubleshooting” on page 101 supplies solutions to problems that could occur while using RMS.
- “Collecting information for advanced troubleshooting” on page 105 describes dump commands for consultants and developers.

4.1 Overview

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


- Clusterwide table
- RMS tree
- Graph


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

The log viewer has search facilities based on the following:

- Keywords
- Severity
- Non-zero exit codes

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

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

 Refer to the section “RMS troubleshooting” on page 101 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.

4.2 Debug and error messages

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

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

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



There are also configuration-specific log files in the log directory. It is recommended that administrators evaluate these if necessary. The name of each log file (normally `<userApplication>.log`) depends on the configuration that was set up using the configuration wizards (RMS Wizard Tools or PCS). Consult the RMS Wizard Tools or PCS online documentation for further information.

The `bmlog` log file can also be useful for problem solving.

4.3 Log file categories

Table 2 identifies and explains the RMS log files contained in `/var/opt/SMAWRms/1og`.

Module	File Name	Contents
Everything (base monitor, generic detector)	<code>switchlog</code>	Operational events, such as resource switches or bugs. Normally, <code>switchlog</code> is the only log file users need to examine.
generic detector	<code><detname>g<n>.log</code>	All messages and job assignments received by the detector. Also contains resource state change information and all error messages. <i>detname</i> is the name of the detector (<code>hvdet_*</code> for RMS Wizard Tools and <code>pcsdet_*</code> for PCS) in the <code><RELIANT_PATH>/bin</code> directory.
base monitor	<code>abortstartlog</code>	This file contains records about <code>bm</code> exit conditions to assist support personnel in determining why RMS failed to start. This file is generated and the following console message appears when startup failures occur: FATAL ERROR: RMS has failed to start! Further details about the problem may appear in the <code>switchlog</code> .

Table 2: Log files

Module	File Name	Contents
base monitor	bmlog	<p>General RMS error and message logging information ranges from simple message reporting to more complete information. The error log level determines the contents of this file, which is specified when the base monitor is started. See “Managing base monitor log output” on page 70 for more information.</p> <p>This file includes all messages received by the base monitor at runtime. It also contains information generated by <code>hvdump</code> (see “Using the <code>hvdump</code> command (RMS)” on page 105).</p> <p>Should be used by experts only, since turning on log level flags consumes a great deal of disk space. By default, RMS places no messages in <code>bmlog</code>.</p>
base monitor	tracelog	<p>Records all messages between objects and all modification instructions. By default, RMS places no messages in <code>tracelog</code>.</p>

Table 2: Log files

4.4 Managing base monitor log output

At runtime, the RMS base monitor can generate log messages in the `RELIANT_LOG_PATH/switchlog` file and in the system log file. This section describes how you can control the **log level** (the amount of detail) in the log files. Each log level reports an internal function that is intended for expert use.

i Executing RMS with several active log levels will affect system performance. This feature should be enabled for testing or debugging purposes only.

4.4.1 Managing base monitor log levels

By default, base monitor logging is turned off when RMS starts up. You can set the initial log level by specifying `hvcn` with the `-l` option:

```
hvcn -l <level> -c <configuration_file> ...
```

Is RMS is already running, you can use the following commands to manage the base monitor log level:

```
hvutil -l display
hvutil -l off
hvutil -l <level>
```

The first `hvutil` command displays the current base monitor log level setting, and the second `hvutil` command turns off base monitor logging.

Specifying '`-l <level>`' with `hvcn` or `hvutil` activates base monitor logging, where `<level>` is one or more numbers in one of the following formats:

- A single number.
- A comma-delimited or space-delimited list. If the list is space-delimited, enclose the list in quotes. Examples: `2,4,5,7` `"2 4 5 7"`
- A single hyphen-separated range in the form `n1-n2`. This includes all log levels from `n1` up to and including `n2`. Specifying `-n2` is the same as `1-n2`, and specifying `n1-` includes all log levels above `n1`. The `n1` value must be greater than or equal to 1. Examples: `2-7` `-7` `2-`

i Specifying a log level of 0 (zero) activates **all log levels**. You must specify the special level `off` to deactivate logging.

The current log level remains in effect until another 'hvutil -l <level>' is issued, a 'hvutil -l off' is issued, or until RMS is shut down.

Table 3 lists the valid base monitor log levels.

Log Level	Meaning
0	Turn on all log levels
1	Unused
2	Turn on detector tracing
3	hvdisk level
4	Turn on mskx tracing (stack tracing of the base monitor)
5	Error or warning message
6	Heartbeats and communication
7	Base monitor level
8	Generic controller message
9	Administrative command message
10	Basic-type level
11	Dynamic reconfiguration contracting level
12	Shutdown debug level
13	Token level
14	Detector message
15	Local queue level
16	Local queue level
17	Script level
18	userApplication contract level
19	Temporary debug traces
20	SysNode traces
21	Message level
22	bm tracelog
23	SatNode trace level

Table 3: Base monitor log levels

4.4.2 Controlling base monitor output to the system log

By default, the RMS base monitor writes the same messages to both the `switchlog` file and the system log file. This behavior is controlled by the `HV_SYSLOG_USE` environment variable.

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

To suppress RMS messages in the system log, edit the `hvenv.local` file and set `HV_SYSLOG_USE=0`. You will have to restart RMS for the change to take effect.

Note: For Log3 RMS messages, the component number is 1080023.

4.5 Managing application log output

Runtime components provided by the RMS configuration tools (PCS and the RMS Wizard Tools) write log messages to files in the same log directory as the RMS base monitor, which is defined in the `RELIANT_LOG_PATH` environment variable. Messages from these components can be broken down into two categories:

- Messages from resource detectors
- All other messages

Detector logging is described in “Managing detector log output” on page 74. The following discussion applies to all other log messages.

The runtime components provided by the configuration tools log everything at an application level. In addition to the normal user-oriented messages, they can also generate debug messages with various levels of detail. The user and debug messages may be written to the following files in `RELIANT_LOG_PATH`:

- `switchlog`—The standard subapplications supplied by PCS and the Wizard Tools record detector reports related to fault and offline transitions into the `switchlog` file.
- `<application_name>.log`—The application-specific log file records all messages associated with that application. The output from any scripts run by the application also go into its log file. The file is created when either offline or online processing for the application begins.

- `hvdet_<xxx>.g<n>log` and `pcsdet_<xxx>.g<n>log`—These are detector log files which record all relevant information regarding the resources they are monitoring, such as all state transitions.

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

4.5.1 Controlling debug messages from standard scripts

You can control debug output from the standard PCS and Wizard Tools scripts by setting the environment variable `HV_SCRIPTS_DEBUG` in the `hvenv.local` file:

- To turn on debug output, create the following entry:

```
export HV_SCRIPTS_DEBUG=1
```
- To turn off debug output, either remove the `HV_SCRIPTS_DEBUG` entry, comment it out, or set the value to 0.

4.6 Managing detector log output

Each instance of an RMS detector writes information to both the `switchlog` file and to its own dedicated log file, which has a name of the form

```
RELIANT_LOG_PATH/<detector_name>.g<n>log
```

where *<detector_name>* is the name of the detector and *<n>* is the instance number of the detector. There is one such file for every running instance of a detector. Note that PCS detectors are named `pcsdet_<xxx>`, and Wizard Tools detectors are named `hvdet_<xxx>`, where *<xxx>* denotes the resource type.

For instance, the log file for the first (and only) `pcsdet_system` detector is `<RELIANT_LOG_PATH>/pcsdet_system.g1log`.


Resource state changes are recorded in both the `switchlog` file and the detector's dedicated log file. All other detector messages are recorded only in the dedicated log file.

4.6.1 Debug messages and log levels

Each detector instance maintains an internal 10 KB memory buffer that it uses to maintain debug messages. These are written to the dedicated log file when a debug event (an unexpected resource status report) occurs. The buffer is circular, so when a new message would cause the buffer to overflow, it will instead overwrite one or more of the oldest stored messages. This continues until a debug event causes the entire buffer to be written to the log file, at which time the buffer is reset. Therefore, only the latest messages will be logged. Some older data may be lost, depending on when a debug event occurs.


When the buffer contents are flushed to the log file, the messages are written in chronological order. It is important to note that the latest message in the buffer was generated by the debug event, but earlier messages in the buffer may or may not be related. Therefore, when you examine the detector log, you may find a series of benign messages, followed by a debug event message, followed by more benign messages, followed by another debug event message, etc. This means you must check the timestamp and text of every detector debug message to see if it pertains to the problem you're trying to resolve.

Every message that the detector can generate has an assigned detail level, where higher detail levels produce more debugging information. At runtime, only those messages with a detail level less than or equal to the detector's current log level will be added to the internal buffer. By default, every detector log level is initially set to 1.

 The amount of information produced by a specific log level depends on the detector resource type. A high log level for one detector may produce less output than a low log level for another detector.

4.6.2 Setting the detector log level

You can set individual log levels for each resource detector from the command line, from PCS, or from the Wizard Tools.

 To enable detector logging, the RMS base monitor log level **must include level 2** (detector tracing), regardless of the individual detector log level settings. By default, the base monitor log level is `off`, so detector logging is suppressed. See “Managing base monitor log output” on page 70.

4.6.2.1 Setting the detector log level with `hvutil -L`

If RMS is already running, you can use the following commands to manage the detector log level:

```
hvutil -L display <resource>
```

```
hvutil -L <level> <resource>
```

The first `hvutil` command displays the current detector log level for the indicated resource, and the second form of the command sets the log level:

- `<level>` can be either 0 (zero) to turn off logging for the resource, or a positive number to turn on logging at that level.
- `<resource>` is the name of the resource monitored by the detector. This must be a generic resource, *i.e.*, it must be an object of type `gResource`. You can find these in the `hvdisp -a` output.

Note that the actual messages generated for the indicated log level and resource will depend on whether the detector was provided by PCS or the Wizard Tools.

The current log level remains in effect until another ‘`hvutil -L`’ command changes the log level, ‘`hvutil -l off`’ is issued, or until RMS is shut down.

4.6.2.2 Setting the detector log level with PCS

PCS detectors recognize log levels in the range 1 to 4. The default value is 1. You can modify the log level for any detector from the PCS GUI:

1. Make sure you are in expert mode by selecting *Option* → *Expert Mode*.
2. Click anything in the left-hand configuration tree and then select *Edit* → *Detector Settings*. The *Detector Settings* view appears (Figure 11).

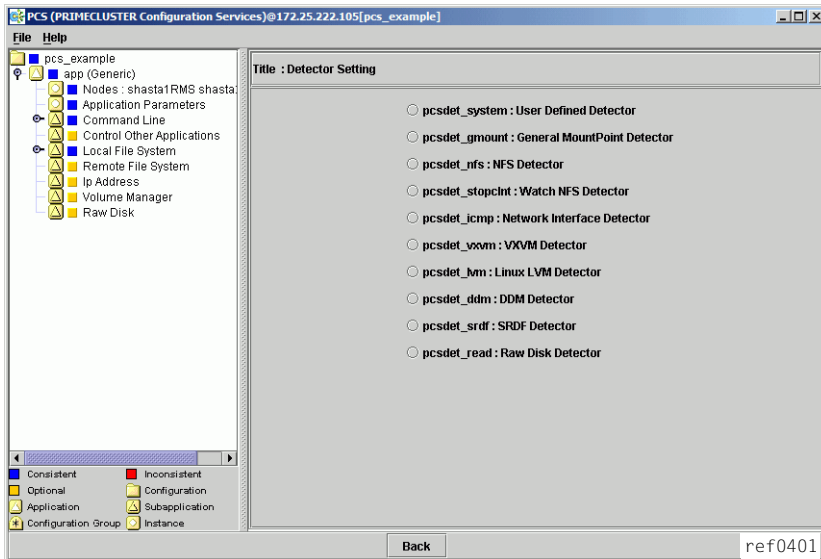


Figure 11: PCS Detector Details menu

3. Click the radio button of the detector you wish to modify, and you will automatically advance to a view that displays the current settings in the right pane. For example, the *pcsdet_system* settings are shown in Figure 12.

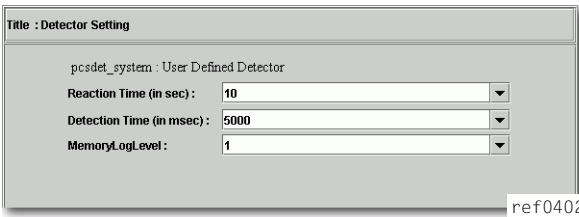


Figure 12: PCS Detector Details menu for pcsdet_system

4. Use the *MemoryLogLevel* drop-down list and change the value to the desired level. The log level applies only to the selected detector.
5. Click *Back* to record the settings and return to the *Detector Settings* view. There is no way to cancel any changes you make in this view.
6. At the *Detector Settings* view, click *Back* to return to the previous view.



Detector log levels set from the PCS GUI apply to the entire configuration and take effect after the configuration is activated and RMS is restarted. The PCS GUI does not dynamically change the runtime log levels.

Runtime adjustment with pcsloglev command

You can make temporary, runtime adjustments to all PCS detector log levels with the `pcsloglev` command:

```
pcsloglev [-a] level
```

level is in the range 0 to 4, where 0 restores all detector logging to the configuration settings. If no level is specified, the current setting is displayed. By default, `pcsloglev` sets the detector log level only on the local node. Use the `-a` option to apply the setting to all nodes in the cluster.



Caution

The `pcsloglev` setting persists across configuration activations, RMS restarts, and node reboots. At the end of your debug session, be sure to issue a `'pcsloglev [-a] 0'` command to restore all detector log levels to their configuration defaults. If `pcsloglev` is set to a value greater than 1, the `RELIANT_LOG_PATH` file system will eventually overflow and lead to a system outage.

4.6.2.3 Setting the log level in the Wizard Tools

The valid range of log levels is 1 to 9, and the default value is 1. The log level can be modified from the `hvw` command as follows:

1. Select the *Configuration-Edit-Global-Settings* menu.
2. Choose the *DetectorDetails* sub-menu. The screen in Figure 13 appears.

```

Detector Details:
 1) HELP                               12) ReactionTimeOfhvdet_nfs=10
 2) RETURN                              13) ReactionTimeOfhvdet_rcvm=33
 3) DEFAULTVALUES                       14) ReactionTimeOfhvdet_read=10
 4) ReactionTimeOfhvdet_ckhost=10       15) ReactionTimeOfhvdet_srdf=31
 5) ReactionTimeOfhvdet_ddm=19          16) ReactionTimeOfhvdet_stopclnt=10
 6) ReactionTimeOfhvdet_execbin=10      17) ReactionTimeOfhvdet_system=10
 7) ReactionTimeOfhvdet_glbassrt=10     18) ReactionTimeOfhvdet_vxvm=30
 8) ReactionTimeOfhvdet_gmount=10       19) ReactionTimeOfForeignDetectors=30
 9) ReactionTimeOfhvdet_icmp=10         20) MemoryLogLevel=1
10) ReactionTimeOfhvdet_locassrt=10     21) DynamicDetectorLogging=0
11) ReactionTimeOfhvdet_lvm=18
Set details for a detector:

```

Figure 13: Wizard Tools Detector Details menu

3. Select *MemoryLogLevel* and change the value to the desired level. The log level applies to all detectors.

Runtime adjustment with the Wizard Tools


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

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

The default value is 0, which turns off debugging. Values in the range 1 to 9 turn on debugging, and higher numbers produce more information. The volume of generated log messages varies according to the detector, so a high value for one detector may produce fewer messages than a low value for another. Changes to this value do not take effect until the next time the configuration is activated.

Using `hvw` to turn on dynamic logging actually creates the file `<RELIANT_LOG_PATH>/etc/wizardLogLevel`, which simply contains the desired debug level as a single-digit ASCII number. You can bypass the `hvw` command and create (or delete) the `wizardLogLevel` file manually, as long as you observe the following rules:

- If the file does not exist, or if the file contains the number 0 (zero), then debugging is turned off.
- If the file exists but is empty, a value of 3 is assumed.
- If the file contains a value in the range 1 to 9, debugging is turned on.

 Turning on debugging should only be done when problems occur. Once the problems are resolved, debugging should be turned off to avoid filling the file system with extraneous information.

4.7 Interpreting RMS log messages

This section describes the format of messages from various RMS components.

4.7.1 switchlog message format

The `RELIANT_LOG_PATH/switchlog` file records RMS events relevant to the user, such as switch requests and fault indications. It contains the following five message types:

1. Informational messages (notices)
2. Warning messages
3. Error messages
4. Fatal error messages
5. Output from scripts run by RMS

Notices, warnings, errors, and fatal errors

The first four categories of messages all appear in this format:

timestamp: (err_code, err_number): msg_type: msg_text: msg_end

There is a colon followed by a space (:) between each field of the message. The fields are as follows:

1. *timestamp* has the format
yyyy-mm-dd hh:mm:ss.xxx
2. (*err_code*, *err_number*) is a two or three letter code denoting the internal component that generated the message, followed by the unique number of the message for that component.
3. *msg_type* is one of the following:
 - NOTICE
 - WARNING
 - ERROR
 - FATAL ERROR
4. *msg_text* is any text generated by the RMS product. This consists of one or more lines of text, each followed by a newline character.
5. *msg_end* is a series four equal signs (====) followed by a newline character. With the terminal newline of the preceding *msg_text* field and the intervening colon-space field delimiter, this appears as a colon, a space, and four equal signs on a separate line in the log file. This uniquely identifies the end of each message, regardless of the number of lines.

For listings of all RMS `ERROR` and `FATAL ERROR` messages (without the leading *timestamp* and trailing *msg_end* fields), see the chapter “Non-fatal error messages” on page 107 and the chapter “Fatal error messages” on page 191.

Script output

The fifth category of messages, script output, follows no specific format. These messages are simply the redirected standard output and standard error streams generated by scripts defined in the RMS configuration. They generally have neither a leading *timestamp* field nor a trailing *msg_end* field.

Here is an example of a short sequence of script messages:

```
Starting Reliant Monitor Services now
loadcount: 0
loadcount lt 3
```

4.7.2 Application log message format

Applications and resources configured by PCS or the Wizard Tools produce log messages in the following format:

```
resource_name: state: timestamp: msg_type: msg_text: msg_end
```

There is a colon followed by a space (:) between each field of the message. The fields are as follows:

1. *resource_name* is the name of the particular resource node in the RMS graph whose script is running. This field may be empty if no resource is associated with the message.
2. *state* is an indication of the type of action that is being performed, and is the value as set by RMS in the environment variable `HV_SCRIPT_TYPE`. The field typically contains the values `online` or `offline`. The RMS configuration tools also set the field with the value `PreCheck`, when a `PreCheck` script is being run. This field will be empty for messages of type `DEBUG` being printed.
3. *timestamp* contains the date and time when the message was generated. It appears in the format `yyyy:mm:dd hh:mm:ss.xxx`, 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.xxx* is the number of seconds and milliseconds past the hour.
4. *msg_type* is one of the following:
 - `DEBUG`
 - `NOTICE`
 - `WARNING`
 - `ERROR`
 - `FATAL ERROR`
5. *msg_text* contains the text generated by the RMS Wizard product. This text may consist of more than one line, *i.e.*, it may have **embedded** newline characters. In general, it is not **terminated** with a newline character.
6. *msg_end* is a series of four equal signs (====). Since *msg_text* is not usually terminated with a newline, *msg_end* appears on the same line.

4.7.3 Program log message format

RMS consists of a number of individual programs that contribute to the overall high availability management. Each of these programs can generate its own trace and error messages in a dedicated log file, which can be found at

```
RELIANT_LOG_PATH/<program_name>log
```

For example, the RMS base monitor program is named `bm`, and its dedicated log file is `RELIANT_LOG_PATH/bmlog`.

Program trace and error messages are intended for internal troubleshooting purposes only. The format is described here in case you are asked to provide the information to a PRIMECLUSTER consultant.

Trace messages have the following prefix:

```
timestamp: file: line:
```

Error messages have the following prefix:

```
timestamp: file: line: ERROR
```

Here is an example of a short sequence of trace messages:

```
2005-12-17 14:42:46.256: det_generic.C: 756: Return from  
GdCheck  
2005-12-17 14:42:46.256: det_generic.C: 773: Call to  
GdGetAttribute  
2005-12-17 14:42:46.257: det_generic.C: 775: Return from  
GdGetAttribute  
2005-12-17 14:42:46.257: det_generic.C: 790: reporting state  
to bm
```


4.8 Viewing RMS log messages

The Cluster Admin interface provides a log viewer that lets you view and filter entries in the RMS switchlog and individual application logs on any node.

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

View the switchlog for a system node as follows:

- ▶ Right-click on the system node and select *View Switchlog* from the pop-up context menu (Figure 14). Alternatively, select a node and use *Tools* → *View switchlog* (Figure 15).

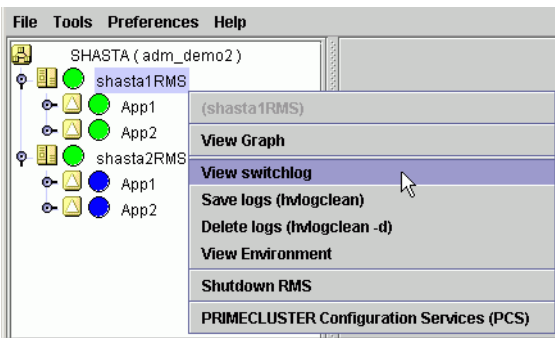


Figure 14: Viewing the RMS switchlog file using a context menu

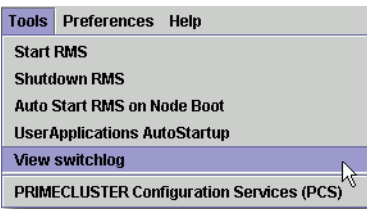


Figure 15: Viewing the RMS switchlog file using the Tools menu

View an application log as follows:

- ▶ Right-click on an application on the RMS tree and choose *View logfile* from the pop-up context menu (Figure 16).

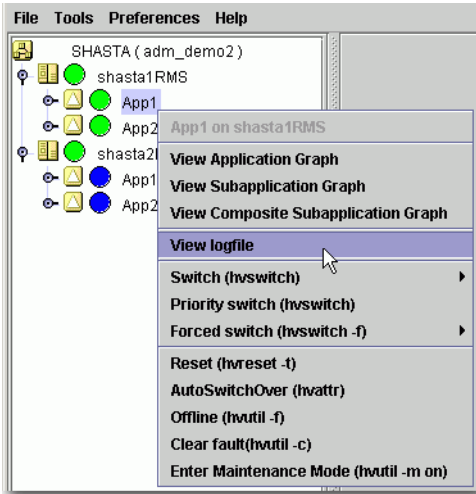


Figure 16: Viewing an application log using a context menu



You can invoke equivalent context menus for an object from the Cluster Admin view, from the clusterwide table, or from any RMS graph containing that object.

By default, each log file is displayed in a separate tab in the right pane (Figure 17).

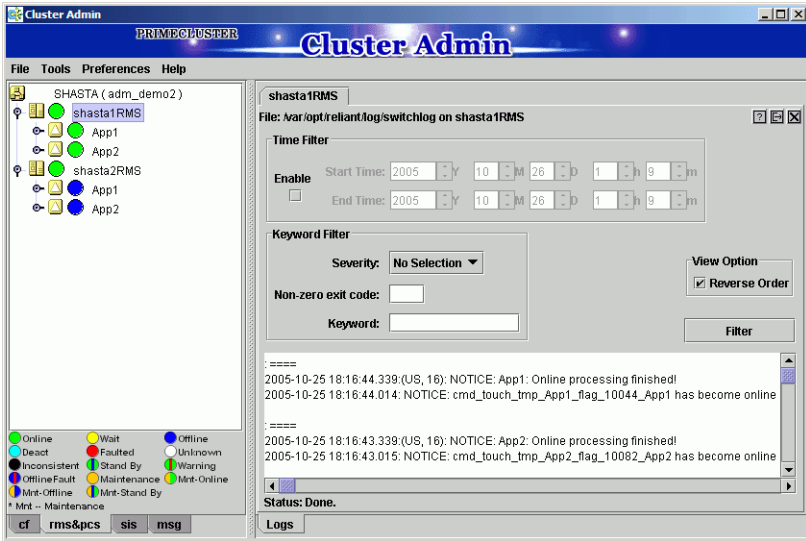


Figure 17: RMS switchlog in tab view

- ▶ To view any tab in a separate window, click the **detach** control button. The detach button is located between the **help** and **close** control buttons in the upper-right corner of the view (Figure 18).



Figure 18: Detail of tab view showing detach button

The detached view contains the same information as the tabbed view (Figure 19).

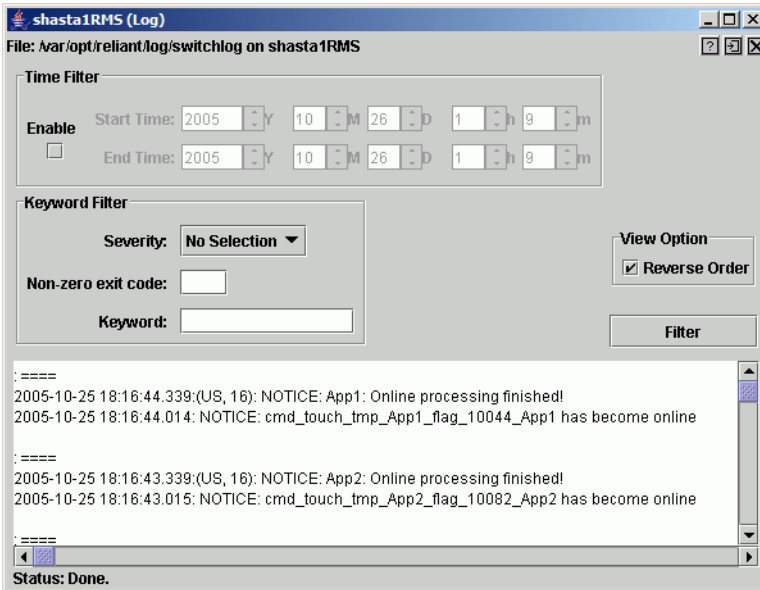


Figure 19: RMS switchlog in detached view

- ▶ To rejoin the detached window to the Cluster Admin view, click the **attach** control button. The attach button is located between the view's **help** and **close** control buttons in the upper-right corner, just below the standard window control buttons (Figure 20).



Figure 20: Detail of detached window view showing attach button

i While in detached mode, the view's close button and the standard window **close** button serve the same purpose: they both close the detached window.

In attached mode, the tabbed view's **close** button closes only the visible tab. All other tabs remain open.

4.8.1 Common procedures for switchlog and application log

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

- *Time Filter*—defines the time period of interest.
- *Keyword Filter*—selects a particular resource name (for an application only), error message severity level, non-zero exit code, or keyword.



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

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



All the selected and non-blank *Time Filter* and *Keyword Filter* controls are combined with a logical AND operation.

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

4.8.2 Time filter

The controls in the *Time Filter* panel allow you to limit the entries displayed in the log pane according to their date and time (Figure 21).

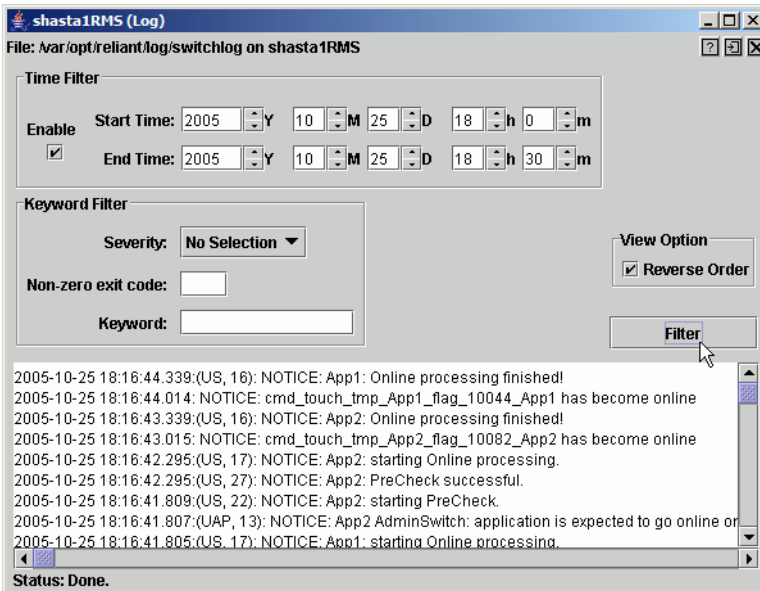


Figure 21: Search based on date and time range

- ▶ Select the *Start Time* and *End Time* using the scrolling input boxes (you can also type in the values directly) and then check the *Enable* checkbox.

The controls take effect the next time you click the *Filter* button.

- ▶ To remove the time filter, uncheck *Enable* and then click *Filter*.

4.8.3 Keyword filters

The following items are available in the *Keyword Filter* panel.

4.8.3.1 Resource Name



The *Resource Name* control is available only for application logs.

- ▶ Select a resource name from the dropdown list (Figure 22) and then click *Filter*.

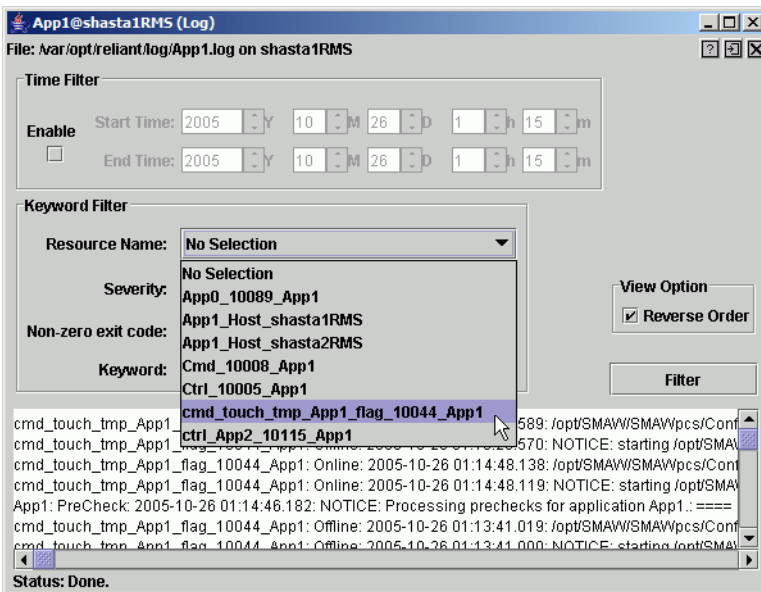


Figure 22: Search based on resource name

- ▶ To remove the resource name filter, select *No Selection* from the dropdown list and then click *Filter*.

4.8.3.2 Severity

- ▶ Select an message severity level from the dropdown list (Figure 23) and then click *Filter*.

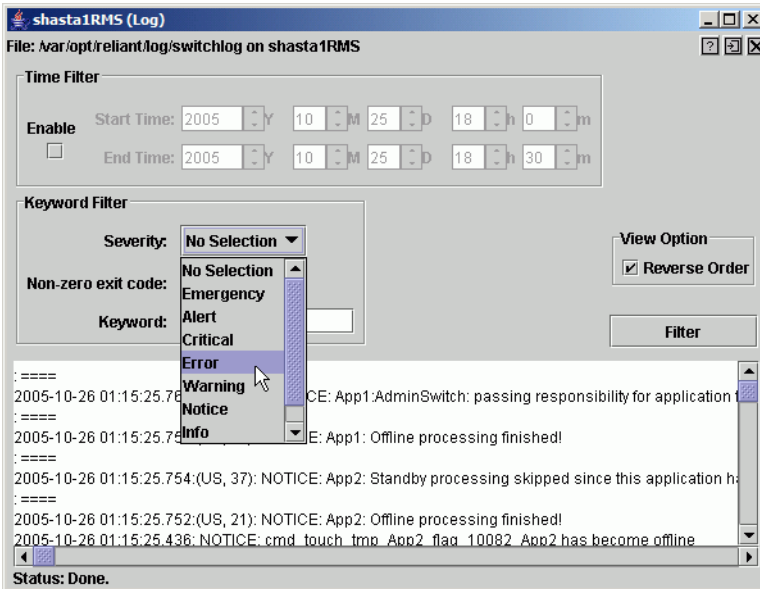


Figure 23: Search based on severity level

Table 4 summarizes the RMS message log viewer severity levels.

Severity level	Description
<i>Emergency</i>	Systems cannot be used
<i>Alert</i>	Immediate action is necessary
<i>Critical</i>	Critical condition (fatal error)
<i>Error</i>	Error condition (non-fatal error)
<i>Warning</i>	Warning condition
<i>Notice</i>	Normal but important condition
<i>Info</i>	Miscellaneous information
<i>Debug</i>	Debug messages

Table 4: RMS severity level description

- ▶ To remove the severity level filter, select *No Selection* from the dropdown list and then click *Filter*.

4.8.3.3 Non-zero exit code

- ▶ Enter a numeric exit code in the *Non-zero exit code* input box and then click *Filter*.

4.8.3.4 Keyword

- ▶ Enter a string in the *Keyword* box (Figure 24) and then click *Filter*.

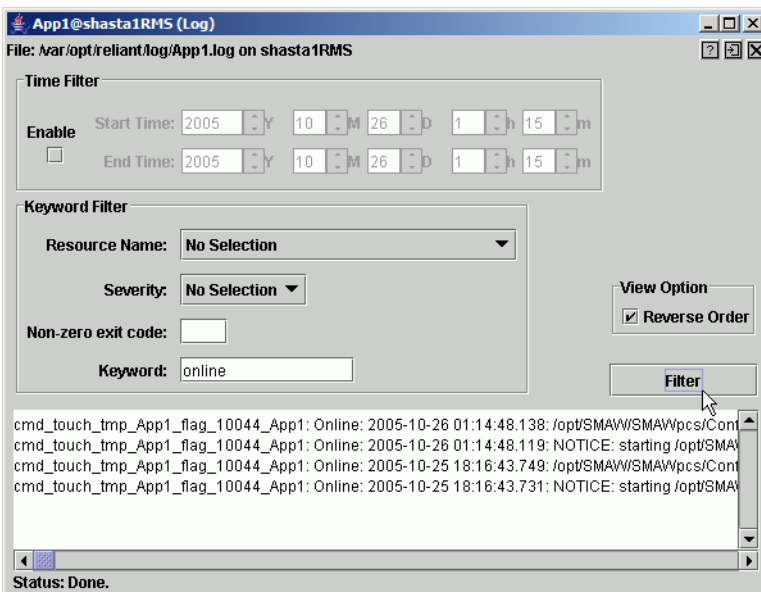


Figure 24: Search based on keyword

i Special characters and spaces are valid, but wildcards are not interpreted. This search is **not case-sensitive**.

- ▶ To remove the keyword filter, clear the text in the *Keyword* box and then click *Filter*.

4.8.4 Text search

You can search the text in the application log by right-clicking on the displayed text. A pop-up dialog with a *Find* entry allows you to perform a **case-sensitive** search in the direction you specify (Figure 25).

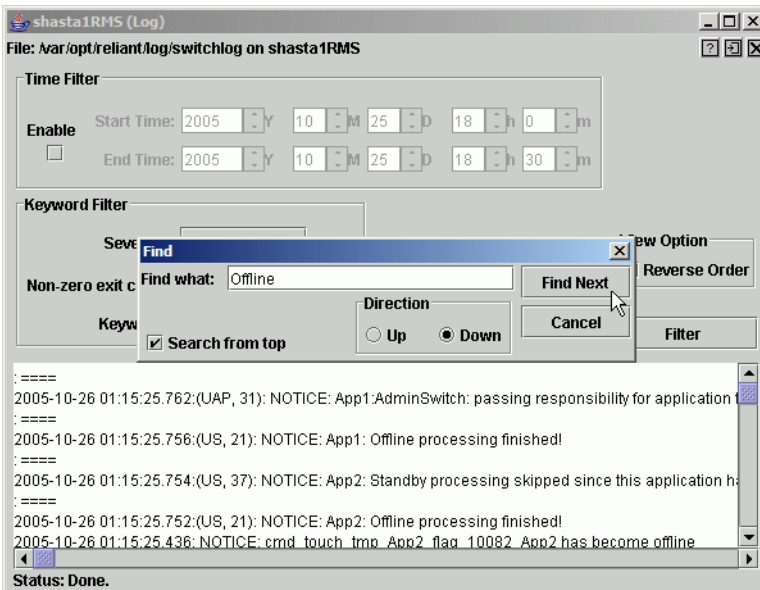


Figure 25: Using the pop-up Find dialog in log viewer



The *Find* search string is processed literally. You can include spaces and special characters, but wildcards are not interpreted.

4.8.5 Removing filters

To remove all filters, take the following steps:

- Uncheck the time filter *Enable* box.
- Set drop-down lists to *No Selection*
- Clear text from input boxes
- Click the *Filter* button

The unfiltered view will be restored.

4.9 RMS log file cleanup

RMS provides two methods for housekeeping of its log files:

1. `hvlogclean` deletes or backs up log files based on elapsed time
2. `hvlogcontrol` issues warnings or deletes files based on the available space in the file system

Both of these are automatically installed and configured with default settings as part of the RMS installation. This section outlines their functions and lists their control settings.

4.9.1 `hvlogclean`

The `hvlogclean` utility saves the current log files into a backup subdirectory of `RELIANT_LOG_PATH` whose name is the time RMS was last started. If invoked with the `-d` option, it will delete the current log files rather than copy them. In either case, `hvlogclean` creates a clean set of log files even while RMS is running.

`hvlogclean` also purges old files in the backup subdirectory. It determines whether or not a backup file is “old” by checking the value of the `RELIANT_LOG_LIFE` environment variable. If the backup file is older (in days) than the number in the variable, the file will be deleted. See “`RELIANT_LOG_LIFE`” on page 307 for more information.



`hvlogclean` is invoked periodically from the `crontab` file, but it can also be invoked manually, and it has a manual page.

4.9.2 `hvlogcontrol`

The `hvlogcontrol` utility prevents log files from consuming all the free space on the file system containing `RELIANT_LOG_PATH`. According to the limits set in the configuration, `hvlogcontrol` can warn the user, delete only the subdirectories in `RELIANT_LOG_PATH`, or delete all RMS log files. See the descriptions of the following RMS environment variables for more information:

- “`HV_LOG_ACTION_THRESHOLD`” on page 306 (global)
- “`HV_LOG_WARN_THRESHOLD`” on page 306 (global)
- “`HV_LOG_ACTION`” on page 309 (local)

Changing these controls requires editing the `hvenv.local` file and then restarting RMS. See “Setting environment variables” on page 303.



`hvlogcontrol` is invoked periodically from the `crontab` file. It has no manual page.

4.9.3 Logging of background scripts

A Command Line subapplication can invoke a script that executes another command or script in the background. If the background process writes to `stdout` or `stderr`, that output will be directed to a log file that remains open until the process ends normally or is terminated. This will cause the log file to be handled differently when RMS log files are cleaned via the `cron` job or `hvlogclean`.

When `hvlogclean` cleans up old log files, it moves them into a backup directory named according to the time RMS was last started. The open log file of the background process will be moved there, too. Because the background process continues to write into the open file, future output for the process will therefore be found in the backup directory, and not in `/var/opt/reliant/log` along with the rest of the RMS processes.

If the log file of the background process is removed while the process is still running (for example, either manually or by the `'hvlogclean -d'` command), the log file will no longer be visible. However, the file will still exist, and the background process will continue to write into the now-invisible file until the process ends or is stopped.

4.10 Configuration troubleshooting

This section describes troubleshooting resources that are available in the standard configuration tools.

4.10.1 PCS log and trace files

Experts and service personnel have two sources of information that may be useful in diagnosing PCS problems:

- The PCS log file is `/var/opt/SMAWpcs/log/pcs.log`. Basic diagnostic reports are written automatically to this file whenever PCS detects an internal inconsistency or error.
- Some problems may not be detected by the PCS self-diagnosis methods. In this case, use *Tools* → *Trace* in the PCS GUI and check one of the trace level checkboxes; higher trace levels generate more information. Trace files are created in the `/var/opt/SMAWpcs/trace` directory.

i Trace information may generate large amounts of output. After a problem has been resolved, turn off tracing by unchecking the *Tools* → *Trace* checkboxes, and remove the unneeded files from the `trace` directory.

Note that the PCS log files mentioned above are all in the `/var/opt/SMAWpcs/` tree and that they are generated at configuration time, not at runtime.

4.10.2 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. Both PCS and the RMS Wizard Tools provide a method to accomplish this for any object in the configuration.

i Testing a script for one resource may require you to first execute scripts for other resources. Therefore, you should have a clear understanding of how RMS normally brings resources online (bottom-to-top in the graph) or takes them offline (top-to-bottom in the graph).

If you use this function at runtime, use it in accordance with the manual of the PRIMECLUSTER Products.

4.10.2.1 PCS GUI method

Use the following procedure to execute a script from the PCS GUI:

- ▶ Make sure the configuration has been activated. Simply generating the configuration is not sufficient.
- ▶ From the PCS menu, select *Tools* → *Draw Graph* to open a graph of the configuration.
- ▶ Right-click on an object in the graph to list the scripts that are available for manual execution (Figure 26).

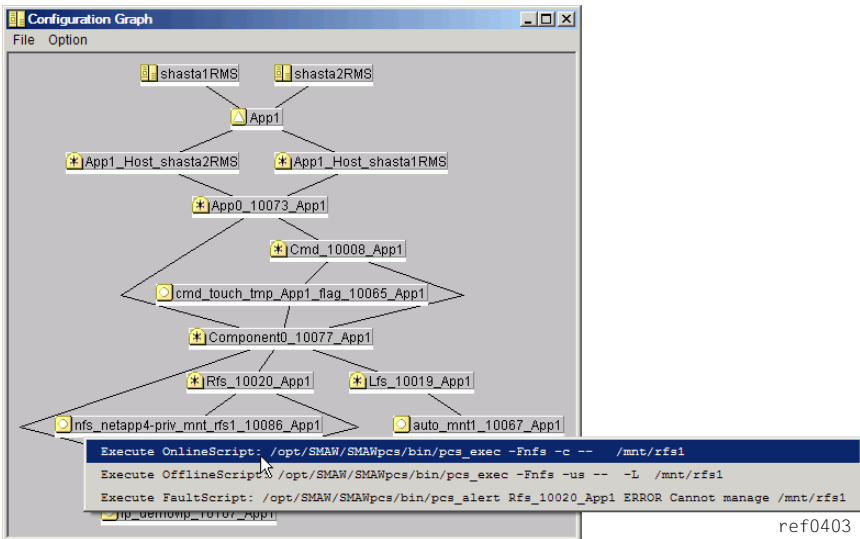


Figure 26: Manually executing a script from a PCS graph context menu

- ▶ Click on the script name in the context menu.
- ▶ If RMS is running, you will be prompted to confirm the procedure before the script is executed.



If you are manually executing a script in a production high availability environment, make sure you understand the possible side effects before you proceed.

The output and return status from all manually executed scripts will appear in the *Manual Script Execution* window (Figure 27).

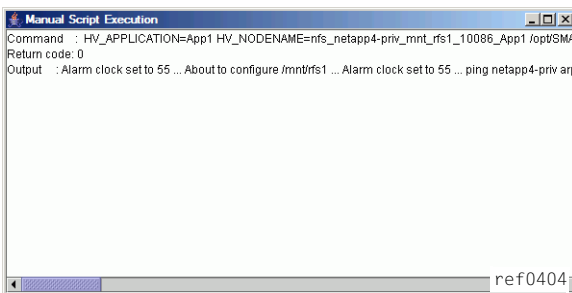


Figure 27: PCS GUI Manual Script Execution output window

Note that you may have to scroll down to see the output from the last execution. Script output is also logged in the file `/var/opt/SMAWpcs/1log/pcs.1log`.

4.10.2.2 PCS CUI method

Use the following procedure to execute a script from the PCS CUI:

- ▶ Make sure the configuration has been activated. Simply generating the configuration is not sufficient.
- ▶ Navigate to the PCS CUI *Configuration Start* screen (Figure 28).

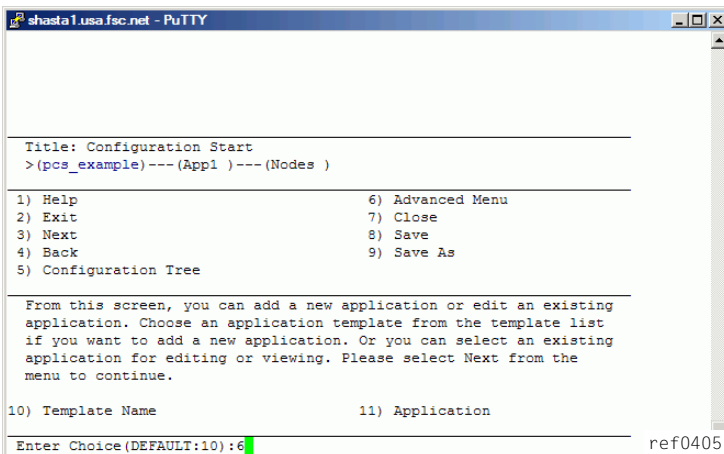


Figure 28: PCS CUI Configuration Start screen

- ▶ Select 6 to display the *Advanced Menu* screen (Figure 29).

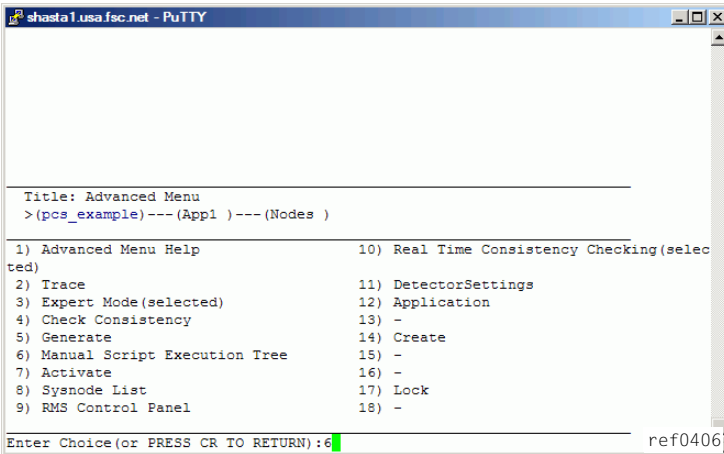


Figure 29: PCS CUI Advanced Menu

- ▶ Select 6 to display the initial *Manual Script Execution Tree* screen (Figure 30).

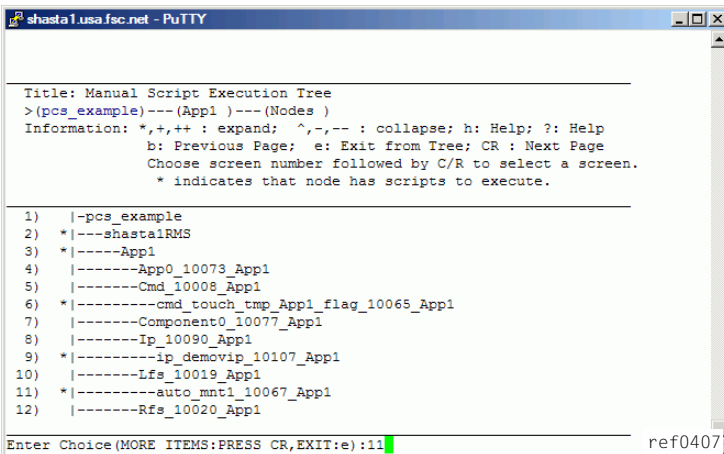


Figure 30: PCS CUI Manual Script Execution Tree—selecting an object

If the phrase *MORE ITEMS: PRESS CR* appears in the prompt at the bottom of the screen, you can press **[Enter]** to advance to another screen that displays additional objects.

- ▶ Enter the number of the object that has the script you want to execute.

In this example, we will select 11, which corresponds to the local file system instance on the first node. Figure 29 displays the list of available scripts for this object.



Figure 31: PCS CUI Manual Script Execution Tree—selecting a script

- ▶ Enter the number of the script you want to execute.

In this example, we will select 1 to execute the *OnlineScript*. Figure 32 displays the results.

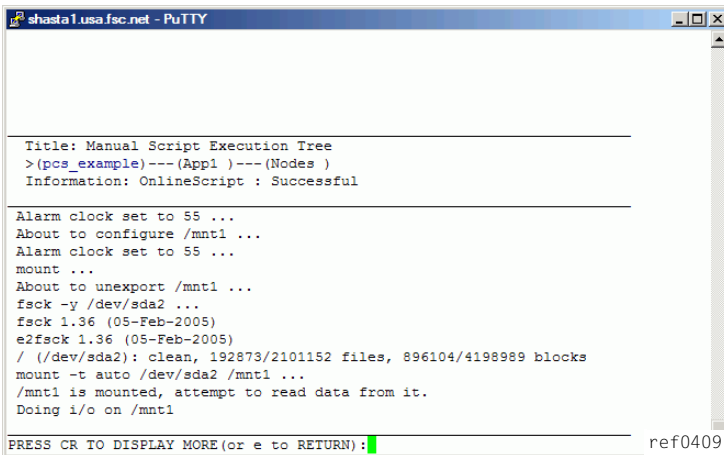


Figure 32: PCS CUI Manual Script Execution Tree—viewing script results

Note that the final status appears on the *Information* line in the upper part of the screen, and the script output appears in the lower part of the screen. The prompt at the bottom of the window will inform you if additional output is available.

Script output is also logged in the file `/var/opt/SMAWpcs/log/pcs.log`.

4.10.2.3 RMS Wizard Tools method

- Navigate to the *Main configuration menu* (Figure 33).

```
shasta2: Main configuration menu, current configuration: test
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 33: Wizard Tools Main configuration menu

- Select 15 for *Configuration-ScriptExecution*.

The *ManualExecution: Application selection menu* opens (Figure 34).

```
ManualExecution: Application selection menu:
 1) HELP
 2) QUIT
 3) TOP
 4) RETURN
 5) app1
Application Name: 5
```

Figure 34: Wizard Tools ManualExecution: Application selection menu

- Select the application name. In this example, we will select 5 for *app1*.

The *ManualExecution: Resource selection menu* opens (Figure 35).

```
ManualExecution: Resource selection menu:
1) HELP
2) QUIT
3) TOP
4) RETURN
5) SHELL
6) Machine000_app1
7) ManageProgram000_Cmd_APP1
8) app1
Choose the resource name: 8
```

Figure 35: Wizard Tools ManualExecution: Resource selection menu

- ▶ Select the resource. In this example, we will select 8 for *app1*.

The *ManualExecution: Script selection menu* opens (Figure 36).

```
ManualExecution: Script selection menu:
1) HELP
2) QUIT
3) TOP
4) RETURN
5) SHELL
6) PreCheckScript:hvexec~-p~app1~test
7) PreOnlineScript:rm~-
f~/usr/opt/reliant/tmp/app1.goingoffline~;~if~[~\ "$HV_INTENDED_STATE\ "~=
~\ "Online\ "~];~then~touch~/usr/opt/reliant/tmp/app1.online;~fi
8) PreOfflineScript:hvenable~app1~ALL~;~rm~-
f~/usr/opt/reliant/tmp/app1.online~;~touch~/usr/opt/reliant/tmp/app1.goingoffline
9) OfflineDoneScript:rm~-f~/usr/opt/reliant/tmp/app1.goingoffline
Choose the script: 6
```

Figure 36: Wizard Tools ManualExecution: Script selection menu

- ▶ Select the script to execute. In this example, we will select 6 for the *PreCheck-Script*

Log output for the selected script will appear in `/var/opt/SMAWRrms/log/<application_name>.log`.

4.11 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 (see “HV_CHECKSUM_INTERVAL” on page 305).
2. There are no applications on this node that are online, waiting, busy, or locked.
3. There are no online remote base monitors encountered by this base monitor.

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

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



Configuration checksum differences often result when global environment variables are changed manually but inconsistently on different nodes.

Action:

To verify that a configuration checksum difference is not the cause of the problem, ensure that all nodes have been updated with the proper configuration by using the following procedure:

1. Stop RMS everywhere in the cluster. This may require an explicit `hvshut` command on any nodes that are not communicating with the others.
2. Determine which configuration to run. Use `hvdisk -a` or `hvdisk -T SysNode` on each node to verify the name of the configuration file. (The `hvdisk` command does not require root privilege.)

A configuration may have the same name but different contents on two or more nodes if one of the following has occurred:

- The most recent activation did not distribute the configuration to all nodes in the cluster. For instance, the user may have specified *Ignore Down Nodes* when activating with PCS.
 - If PCS or the Wizards Tools were used on more than one node in the cluster, there may be more than one version of the configuration with the same name.
3. Activate the correct configuration with the same tool (PCS or Wizard Tools) that was used to create it. For the correct procedure, see the *RMS Configuration and Administration Guide* for PCS or the Wizard Tools.

Alternatively, redistribute the existing `<configname>.us` file with one of the following methods:

- In the RMS Wizard Tools, use *Configuration Push*.
- In PCS, use *Tools -> Activate -> Update*.

Make sure the activation is successful so that all nodes are updated.

4. Start RMS on the entire cluster. All nodes should now be running the same configuration.
- RMS hangs after startup (processes are running, but `hvdisk` hangs)

This problem might occur if the local node is in the CF state LEFTCLUSTER from the point of view of one or more of the other nodes in the cluster.

Action:

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

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

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

This problem could occur if the CIP configuration file `/etc/cip.cf` contains entries for the netmask. These entries are useless (not evaluated by CIP). From the RMS point of view these entries cannot be distinguished from IP

addresses, which have the same format, so RMS will invoke a `gethostbyaddr()`. This normally does no harm, but in some unusual cases the OS may become confused.

Action:

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

Remove the netmask entries, and restart RMS.

- RMS detects a 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.



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, use `'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 RMS base monitor detects a loss of detector heartbeat, but there is no indication as to the reason for the loss.

RMS automatically invokes a tool that provides diagnostic information for this event. The diagnostic tool performs the following actions:

- Invokes `truss(1)` on Solaris or `strace(1)` on Linux to trace the detector process
- Turns on full RMS and detector logging with the `-l0` (lowercase “L”, zero) option
- Gathers system and users times for the process

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

Note that user-specified operations such as detector tracing will continue on each node, even if it appears to have left the cluster. In extreme cases, turning on high detail trace levels may affect the performance of a node and contribute to delays in its base monitor heartbeat.

Action:

See the `switchlog` for the diagnostic information.

4.12 Collecting information for advanced troubleshooting

The procedures in this section provide advanced debugging information and are intended for use by consultants and developers.

4.12.1 Using the `hvdump` command (RMS)

The `hvdump` command collects RMS debugging information on the local node. A normal RMS installation creates the file

`RELIANT_PATH/bin/hvdump`

By default, `RELIANT_PATH` is `/opt/SMAW/SMAWRrms/`. Symbolic links to the same file are also created in `/usr/bin/hvdump` (which is usually in the default search path) and `/opt/SMAW/bin/hvdump`.

Invoking `hvdump` gathers log files and configuration data and places them in the `RELIANT_PATH` directory in the following archive:

`<nodename>RMS.<timestamp>.debug_information.tar.<suf>`

where the suffix `<suf>` is 'gz' on Linux platforms, and 'Z' on Solaris platforms.

The archive file is intended for use by RMS support specialists. The `hvdump(1M)` manual page presents a detailed list of the files and information included in the archive.

5 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
- QUE: Message queues
- SCR: Scripts
- SWT: Switch requests (hvswitch command)
- SYS: SysNode objects
- UAP: userApplication objects
- US: us files
- WLT: Wait list
- WRP: Wrappers

5.1 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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

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

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.

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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

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

The checksum reported by the remote host is different from that of the local host, and the configuration for the remote host `<hostname>` includes the local host, but the configuration for the local host does not include the remote host's name.

Action:

Make sure that the local and the remote host are running the same configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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.

The checksums of the configurations of the local and the remote host are different, and 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 120 seconds otherwise.

Action:

Make sure that all the hosts in the cluster are running the same configuration file. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

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

RMS was 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` 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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

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

The '`hvutil -u`' command has been invoked for a SysNode that is not in the Wait state (internal option).

Action:

Reissue the '`hvutil -u`' command after the node has reached the Wait state.

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

One of the local environment variables `<envattribute>` is not set in `hvenv`. This will eventually cause RMS to exit with exit code 1.

`<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`.

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 SysNode that 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 `<userapplication>` is locked or busy, modification request skipped.

The application is busy. Some other modification is already in progress, or some requests are being processed, or application contracting is ongoing.

Action:

Reissue 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 in the switchlog prior to this message to find out the exact cause of the failure and correct the problem.

- (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 `<file>`.

A error occurred while transferring file `<file>` 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 limit. The timeout limit is the greater of the environment variable `MODIFYTIMEOUTLIMIT` (if defined) or 0. If the variable is not defined, the default timeout limit is 120 seconds.

Action:

Contact field support.

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

The time taken for dynamic modification during bm startup was greater than the timeout limit. The timeout limit is the greater of the environment variable `MODIFYTIMEOUTLIMIT` (if defined) or 0. If the variable is not defined, the default timeout limit is 120 seconds. RMS exits with exit code 63.

Action:

Contact field support.

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

Critical internal error.

Action:

Contact field support.

- (ADC, 37) 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 `<userapplication>`; switch the application offline before making this resource critical.

During dynamic modification, there was an attempt to change a non-critical resource `<resource>` to a critical resource by setting its `MonitorOnly` attribute to 0. The resource was not online, but its parent application `<userapplication>` was online, so dynamic modification was aborted.

Action:

Switch the `userApplication` offline before making the resource critical.

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

If RMS finds that the `userApplication <userapplication>` will have no children while performing dynamic modification, 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 `hvmmod` has been invoked. If this variable cannot be set with the function `putenv()`, RMS generates this error.

Action:

Check the reason `<failurereason>` in the switchlog to find out why this operation has failed and take corrective action.

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

Wizards make use of an action file during `hvmmod`. The execution of this action file (`<command>`) failed due to the process exiting by using an exit call.

Action:

Check the switchlog to find the reason for this failure and take corrective action before reissuing the `hvmmod` 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 `<errornumber>` - `errortext`. 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. The specific reason for this failure is returned by the OS in the error code `<errornumber>` and in the explanation `<errortext>`.

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. A byte count discrepancy indicates the transfer operation was 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*.

The file *<filename>* that has been specified as the file to be copied from the local host to the remote host could not be opened for reading.

Action:

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

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

During a file transfer between the hosts, RMS encountered a problem indicated by the OS error code *<errornumber>*.

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 *<errornumber>*, 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, the *hvdisp* process is restarted automatically. Note that some data may be lost and not displayed at this time. The specific OS error code for the error encountered is displayed in *<errornumber>*.

Action:

Make sure that the host conditions are such that the temporary file can be checked. 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 in the switchlog contains a detailed explanation about the error that occurred 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 use the command `<command>` to prepare its own configuration for a subsequent transfer to the remote host. 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 use the command `<command>` to store remote configuration files for a subsequent dynamic modification on this host. 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 `<hostname>`.

While performing RMS cluster-wide shutdown, RMS on host `<hostname>` 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 <errornumber> while reading file <file>, reason: <errortext>.

While reading file <file>, an error <errornumber> occurred as explained by <errortext>. File reading errors may occur during dynamic modification, or during hvjoin operation.

Action:

Contact field support.

- (ADC, 68) Error <errornumber> while opening file <file>, reason: <errortext>.

While opening file <file>, an error <errornumber> occurred as explained by <errortext>. 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.

5.2 ADM: Admin, command, and detector queues

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

During dynamic modification, when new resource(s) cannot be brought offline before they are added to an offline parent object, RMS generates this error.

Action:

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

- (ADM, 4) 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 `hvmmod` command.

- (ADM, 5) 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`), dynamic modification is aborted.

Action:

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

- (ADM, 6) 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, dynamic modification is aborted.

Action:

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

- (ADM, 7) 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, dynamic modification is aborted.

Action:

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

- (ADM, 8) 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, dynamic modification fails.

Action:

Get rid of the cycles.

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

Deleting a resource causes all its children with no other parents to be deleted as well. Therefore, deleting a resource and then attempting to modify the attributes of the deleted resource, or a child of that resource that has no other parents, causes dynamic modification to fail.

Action:

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

- (ADM, 11) 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) 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) 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, dynamic modification is aborted. Theoretically this is not possible.

Action:

Contact field support.

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

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, dynamic modification will be aborted.

Action:

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

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

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

Action:

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

- (ADM, 16) 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.

Action:

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

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

An attempt was made to link a parent `<parentobject>` and a child `<childobject>` that are already linked, and dynamic modification is 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) 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, dynamic modification is aborted.

Action:

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

- (ADM, 19) 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.

Action:

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

- (ADM, 20) 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 state, is prohibited. 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) 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) 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) 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, dynamic modification is aborted. A subsequent FATAL message is also generated in the switchlog with more details as to why the sanity check failed.

Action:

Make sure that the configuration passes each of the above sanity checks.

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

After an object `<object>` is deleted from the RMS resource graph, attempting to unlink it from its parent object (or vice versa) will cause dynamic modification to fail.

Action:

Make sure that the operations of unlinking and deleting an object are performed in the proper sequence.

- (ADM, 25) 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 this is not the case, dynamic modification is aborted.

Action:

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

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

When a new object is added to an existing configuration, but its parent object *<parentobject>* is not a resource, dynamic modification is aborted.

Action:

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

- (ADM, 27) 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) 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, dynamic modification is aborted.

Action:

Make sure that the child object specified is a resource.

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

Internal error.

Action:

Critical error. Contact field support.

- (ADM, 30) 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, dynamic modification is aborted.

Action:

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

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

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

Action:

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

- (ADM, 32) 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 is aborted.

Action:

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

- (ADM, 33) 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, dynamic modification fails.

Action:

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

- (ADM, 34) 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) 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 in the online state and the other of which is in the offline or standby state, dynamic modification is aborted. 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) Dynamic modification failed: existing parent resource `<parentobject>` is in state `<state>` but needs to be in one of `stateOnline`, `stateStandby`, `stateOffline`, `stateFaulted`, or `stateUnknown`.

During dynamic modification, if the state `<state>` of a parent resource `<parentobject>` is not one of the states `stateOnline`, `stateOffline`, `stateFaulted`, or `stateUnknown`, dynamic modification aborts.

Action:

Make sure that the state of the parent resource is one of the states mentioned above.

- (ADM, 39) Dynamic modification failed: new resource `object` which is a child of application `<userapplication>` has its `HostName` `<hostname>` the same as another child of application `<userapplication>`.

When a new object `<object>` is being added as a child of `<userapplication>` and the value of its `HostName` attribute is the same as the value of the `HostName` attribute of an existing child of `<userapplication>`, dynamic modification is aborted.

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 `<userapplication>`.

- (ADM, 40) Dynamic modification failed: a new child *<childobject>* of existing application *<userapplication>* does not have its `HostName` set to a name of any sysnode.

When a new child object *<childobject>* is added to an application *<userapplication>* during dynamic modification, if the `HostName` attribute is missing for this object, dynamic modification is aborted.

Action:

The first level object under *<userapplication>* must have a `HostName` attribute.

- (ADM, 41) 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 that is supposed to be online, then dynamic modification is aborted.

Action:

Make sure that the parent and the child are in a similar state.

- (ADM, 42) 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 is aborted.

Action:

Make sure that the parent and the child are in a similar state.

- (ADM, 43) 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>*, dynamic modification is aborted.

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) 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) 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) Dynamic modification failed: linking the same resource *<object>* to different applications *<userapplication1>* and *<userapplication2>*.

RMS received a directive to add a new child object *<object>* by linking it to parent objects belonging to different applications *<userapplication1>* and *<userapplication2>*. Dynamic modification is aborted.

Action:

When adding a new child resource, make sure that its parent objects do not belong to different applications.

- (ADM, 47) 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, unless they have 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) Dynamic modification failed: child object *<childobject>* belongs to a deleted application.

An attempt was made to delete an object *<childobject>* that belongs to a deleted application, but deleting the application deleted all its children including *<childobject>*.

Action:

Make sure that before an object is deleted, it does not belong to an application that is being deleted.

- (ADM, 49) Dynamic modification failed: deleted object *<objectname>* belongs to a deleted application.

An attempt was made to delete an object *<objectname>* that belongs to a deleted application, but deleting the application deleted 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) Dynamic modification failed: cannot delete object *<object>* since it is a descendant of a new object.

RMS received a directive to delete an object *<object>* that is a descendant of a new object. Dynamic modification was aborted.

Action:

Make sure that when an object is being deleted, it is not a descendant of a new object.

- (ADM, 51) Dynamic modification failed: cannot link to child *<childobject>* since it will be deleted.

RMS received a directive to link to a child *<childobject>* that is going to be deleted. Dynamic modification was aborted.

Action:

Do not link to a child object that is to be deleted.

- (ADM, 52) Dynamic modification failed: cannot link to parent *<parentobject>* since it will be deleted as a result of deletion of object *<object>*.

There was an attempt to delete an object *<object>* and use its descendant (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. Dynamic modification was aborted.

Action:

Do not attempt to delete an object and use its descendant as the parent for a new resource.

- (ADM, 53) Dynamic modification failed: *<node>* is absent.

An attempt was made to modify the attribute of an object *<node>* that is absent, and dynamic modification is aborted.

Action:

Modify the attributes of an existing object.

- (ADM, 54) Dynamic modification failed: NODE *<object>*, attribute *<attribute>* is invalid.

When RMS receives a directive to modify an object *<object>* with attribute *<attribute>* that has an invalid value, dynamic modification is aborted.

Action:

Specify a valid value for the attribute *<attribute>*.

- (ADM, 55) Cannot create admin queue.

RMS uses Unix queues internally for interprocess communication. In particular, the admin queue is used for communication between RMS and other utilities like hvutil, hvmod, hvshut, hvswitch and hvdisp. If RMS cannot create this queue, it exits with exit code 50.

Action:

Restart RMS.

- (ADM, 57) hvdisp - open failed - *filename*.

RMS was unable to open the file */opt/SMAW/SMAWRms/locks/.rms.<pid>* for writing when hvdisp was invoked.

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 field support.

- (ADM, 58) `hvdisp - open failed - filename: errortext`.

The `hvdisp` command was unable to open the file `<file>` (`/opt/SMAW/SMAWRrms/locks/.rms.<pid>`) for writing.

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 field support.

- (ADM, 59) *userapplication*: modification is in progress, switch request skipped.

Commands like `hvswitch`, `hvutil`, and `hvshut` cannot run in parallel with a non-local `hvmmod`.

Action:

Make sure that `hvmmod` is not operating on `<userapplication>` before `hvswitch` is performed.

- (ADM, 60) `<resource>` is not a `userApplication` object, switch request skipped!

To perform a switch, `hvswitch` requires a `userApplication` as its argument, but the resource `<resource>` is not a `userApplication`.

Action:

Check the `hvswitch` man page for usage information.

- (ADM, 62) The attribute `<ShutdownScript>` may not be specified for object `<object>`.

The attribute `ShutdownScript` is a hidden attribute of a `SysNode`. The RMS base monitor automatically defines its value, and users cannot change it in any way.

Action:

Do not attempt to change the built-in value of the ShutdownScript attribute. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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] <userapplication> [<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., one that appears in the `<configname>.us` file.

- (ADM, 67) `sysnode` Cannot shut down.

The `hvshut -a` command was invoked but not all of the nodes replied with an acknowledgement.

Action:

Login to the remote hosts. If RMS is still running, perform `hvutil -f <userapplication>` to shut down each application one at a time. If this fails, refer to the `switchlog` and `<userapplication>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 field support.

- (ADM, 70) NOT ready to shut down.

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 or dynamic reconfiguration, has terminated.

- (ADM, 75) Dynamic modification failed: child *<resource>* of userApplication object *<userapplication>* has HostName attribute *<hostname>* common with other children of the same userApplication.

The RMS internal sanity-check functions detected 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) Dynamic modification failed: cannot delete object *object* since its state is currently being asserted.

Dynamic modification was attempted on an object that is being asserted.

Action:

Perform the modification after the assertion has been fulfilled.

- (ADM, 78) Dynamic modification failed: PriorityList *<prioritylist>* does not include all the hosts where the application *<userapplication>* may become Online. Make sure that PriorityList contains all hosts from the HostName attribute of the application's children.

Set PriorityList for *<userapplication>* 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) Dynamic modification failed: PriorityList <prioritylist> includes hosts where the application <userapplication> may never become 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 <userapplication> 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) Dynamic modification failed: application <userapplication> may not have more than <maxcontroller> parent controllers as specified in its attribute MaxControllers.

The application <userapplication> uses more parent controllers than allowed by the attribute MaxControllers, which is currently set to <maxcontroller>. Dynamic modification is aborted.

Action:

Make sure that the number of parent controllers used by an application is no more than the number specified as part of the MaxControllers attribute, or modify MaxControllers to increase the number.

- (ADM, 82) Dynamic modification failed: cannot delete *type* <object> unless its state is one of Unknown, Wait, Offline or Faulted.

There was an attempt to delete a SysNode or SatNode from a running configuration, but the node was not in one of the states Unknown, Offline, Wait, or Faulted.

Action:

Shut down RMS on that host and then try the deletion again.

- (ADM, 83) 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 the dynamic modification file does not contain 'delete <sysnode>;' where <sysnode> is the name of the local node.

- (ADM, 84) Dynamic modification failed: cannot add SysNode <sysnode> since its name is not valid.

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) Dynamic modification failed: timeout expired, timeout symbol is <symbol>.

The time required for a dynamic modification exceeded the maximum time allowed.

Action:

Make sure that the network connection between the hosts is functional, and look for the following possible problems:

- The scripts from newly added resources take too much time to execute.
- Dynamic modification adds too many new nodes.

The dynamic modification file is too big or too complex.

- (ADM, 86) Dynamic modification failed: application <userapplication> cannot be deleted since it is controlled by the controller <controller>.

A controlled application <userapplication> cannot be deleted while its controller <controller> retains the application's name in its Resource attribute.

Action:

Do one of the following:

- Remove the name of the deleted application from the controller's Resource attribute
- Add a new application with the same name.
- Delete the controller together with its controlled application.

Change the controller's NullDetector attribute to 1.

- (ADM, 87) Dynamic modification failed: only local attributes such as ScriptTimeout, DetectorStartScript, NullDetector or MonitorOnly can be modified during local modification (hvmod -l).

Only the modification of local attributes is allowed during local modification.

Action:

Make a non-local modification, or modify different attributes.

- (ADM, 88) Dynamic modification failed: attribute *<attribute>* is modified more than once for object *<object>*.

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:

Limit the number of modifications in the file to one per object.

- (ADM, 89) 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) 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.

The user tried to change the Resource attribute of controller `<controllernode>`, but one or more applications in `<oldresource>` are going to be deleted.

Action:

Make deleted applications are not referred to from any controller.

- (ADM, 91) Dynamic modification failed: controller `<controller>` has its Resource attribute set to `<resource>`, but application named `<userapplication>` is going to be deleted.

The user tried to modify the controller `<controller>`, but its Resource attribute `<resource>` contains the application `<userapplication>`, which 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.

RMS was started with the `-a` option but due to an internal error RMS could not be started on the remote host. Critical error.

Action:

Contact field support. For a temporary workaround, try the same command again, or start RMS manually on each host.

- (ADM, 96) Remote startup of RMS failed `<startupcommand>`. Reason: *errortext*.

RMS could not be started on remote hosts because the command `<startupcommand>` failed. This may occur when some of the hosts are not reachable or the network is down.

Action:

Make sure the remote nodes are accessible over the network and then check their status. If any of the nodes are unreachable, take corrective action.

- (ADM, 98) Dynamic modification failed: controller *<controller>* has its Resource attribute set to *<resource>*, but some of the controlled applications from this list do not exist.

The list of controlled applications for controller *<controller>* includes one or more applications that are not running on the local host.

Action:

Correct your modification file so that the controllers refer only to the existing application.

- (ADM, 99) 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.

The user has tried to modify the controller *<controller>* to change its Resource attribute from *<oldresource>* to *<newresource>*. The operation failed due to one of the following reasons:

- One or more of the applications listed in *<newresource>* is not an existing application.
- An application state is incompatible with the state of the controller.
- The application 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) Dynamic modification failed: because a controller *<controller>* has AutoRecover set to 1, its controlled application *<userapplication>* cannot have PreserveState set to 0 or AutoSwitchOver set to 1.

If an application is controlled by a controller that has its AutoRecover attribute set to 1, then the application's attributes must be set as follows: PreserveState=1, AutoSwitchOver=0.

Action:

Check the PreserveState and AutoSwitchOver attributes of the controlled 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 <userapplication> is *length*. This exceeds the maximum allowable length which is *maxlength*.

The cumulative length of the SysNode names specified in the configuration for application <userapplication> exceeds the maximum allowable length.

Action:

Limit the length of the SysNode names so that the list fits is no longer than the maximum allowable length.

- (ADM, 118) Dynamic modification failed: cannot add SatNode <satnode> since its rKind <rkind> is not consistent with the rKind of the other SatNodes.

The rKind of SatNode <satnode> is not the same as the rKind of the other SatNodes.

Action:

Make sure that the rKind of all the SatNodes is the same.

- (ADM, 125) Dynamic modification failed: The <attr> entry <value> for SysNode <sysnode> matches the <attr> entry or the SysNode name for another SysNode.

The entry <attr> must be unique.

Action:

Ensure that the `<attr>` entry is unique.

- (ADM, 126) *childapp*: This application is controlled by controller `<controller>`. That controller is defined as a Local controller, so switching this application must be done by switching the controlling application `<parentapp>`.

An attempt was made to use `hvswitch` with an application that is controlled by a Local controller. Local-mode controlled applications are under the full control of their controlling application, so independent `hvswitch` operations are not allowed.

Action:

To switch the local-mode controlled application, you must switch the controlling application.

5.3 BAS: Startup and configuration errors

- (BAS, 2) Duplicate line in `hvgdstartup`.

RMS detected a duplicate line in `hvgdstartup`. RMS exits 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, there was an attempt to redefine the kind for the `DetectorStartScript`.

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 **Script-Timeout** 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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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 userApplication objects. RMS applications cannot have common objects.

Action:

Redesign your configuration so that no two applications have common objects. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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 a shorter rName. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BAS, 18) ERROR IN CONFIGURATION FILE: The duplicate line number is *<linenumber>*.

RMS detected a duplicate line in the hvgdstartup file.

Action:

Make sure that file hvgdstartup has no duplicate lines.

- (BAS, 19) ERROR IN CONFIGURATION FILE: The NoKindSpecifiedForGdet is *<kind>*, so no kind specified in hvgdstartup.

The kind *<kind>* has not been specified for the generic detector in the hvgdstartup file.

Action:

Specify the kind for the generic detector in hvgdstartup.

- (BAS, 23) ERROR IN CONFIGURATION FILE: DetectorStartScript for object *object* is not defined. Objects of type *type* should have a valid DetectorStartScript attribute.

Object *<object>* does not have its DetectorStartScript defined.

Action:

Make sure that the DetectorStartScript is defined for object *<object>*. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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).

The ScriptTimeout value is less than the detector cycle time. This will cause the resource to appear faulted when being brought online or offline.

Action:

Set ScriptTimeout to a value 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. This message indicates 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.

The configuration is checked before applying a dynamic modification. In this case, the check failed, so the dynamic modification did not proceed.

Action:

Make sure that online objects of type `and` have all their children in online states applying dynamic modification. Use only standard PRIME-CLUSTER configuration tools to create and maintain configurations.

- (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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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>`. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BAS, 31) ERROR IN CONFIGURATION FILE: Failed to load a detector of kind `<kind>`.

A detector could not be started by the RMS base monitor.

Action:

Make sure the detector executable is present in the right location and has execute 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>*.

- (BAS, 37) ERROR IN CONFIGURATION FILE: The *satApplication <satapp>* has no children, or its children are not valid resources.

The *satApplication <satapp>* has no children or they are not valid resources.

Action:

Ensure that *<satapp>* has valid resources as its children.

- (BAS, 38) ERROR IN CONFIGURATION FILE: The *satService <satserv>* has no children, or its children are not valid resources.

The *satService <satserv>* has no children or they are not valid resources.

Action:

Ensure that *<satserv>* has valid resources as its children.

- (BAS, 39) ERROR IN CONFIGURATION FILE: The gResource `<gresource>` belongs to a satellite configuration and it is not allowed to have any children.

The gResource `<gresource>` may not have any children.

Action:

Ensure that `<gresource>` does not have any children.

- (BAS, 40) ERROR IN CONFIGURATION FILE: The object `<object>` belongs to more than one satApplication, `satapplication1` and `satapplication2`. Objects must be children of one and only one satApplication object.

An object was encountered as a part of more than one satApplication. RMS satApplication objects cannot share the same objects.

Action:

Redesign your configuration so that no two applications have common objects.

- (BAS, 41) ERROR IN CONFIGURATION FILE: The object `<object>` belongs to more than one satService, `satservice1` and `satservice2`. Objects must be children of one and only one satService object.

An object was encountered as a part of more than one satService. RMS satService objects cannot share the same objects.

Action:

Redesign your configuration so that no two satService objects share the same objects.

- (BAS, 42) ERROR IN CONFIGURATION FILE: The gResource `<gresource>` belonging to a satellite configuration does not have its HostName attribute set. Please ensure that the HostName attribute is set for this gResource.

The gResource `<gresource>` does not have its HostName attribute set.

Action:

Ensure that the HostName attribute is set for all gResource objects belonging to a satellite configuration.

- (BAS, 43) ERROR IN CONFIGURATION FILE: The object *object* has both attributes "MonitorOnly" and "NonCritical" set. These attributes are incompatible; only one of them may be used.

Both the `MonitorOnly` and `NonCritical` attributes 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>*.

5.4 BM: Base monitor

- (BM, 3) Usage: *progname* [-c *config_file*] [-m] [-h *time*] [-l *level*] [-n]

An attempt has been made to start RMS in a way that does not conform to its expected usage. This message is printed to the switchlog indicating the arguments, and RMS exits with exit code 3.

Action:

Start RMS with the right arguments.

- (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) Dynamic modification failed: line *linenumber*, cannot build an object of unknown type *<symbol>*.

An object of unknown type was added during dynamic modification.

Action:

Use only objects of known types in configuration files. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 18) 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) Dynamic modification failed: line *linenumber*, cannot modify attribute *<attribute>* of object *<object>* with value *<value>*.

An invalid attribute is specified for modification.

Action:

Modify only valid attributes.

- (BM, 20) 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. Only objects of a "user" type can be dynamically modified.

Action:

Use only valid resource types when adding new objects to configuration.

- (BM, 21) 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. Only objects of a “user” type can be deleted.

Action:

Delete only objects that are valid resource types.

- (BM, 23) 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 *<userapplication>* to which *<controller>* belongs.

The *PriorityList* of the controlled application *<controlleduserapplication>* is different from the content of the *PriorityList* of the parent application *<userapplication>* to which the *Follow* controller *<controller>* belongs.

Action:

Make sure that the *PriorityList* of both the controlling application and the controlled application are the same.

- (BM, 24) Dynamic modification failed: some resource(s) supposed to come standby failed.

During dynamic modification, an attempt was made to add new resource(s) to a resource that was in standby mode, but the resources could not also be brought into standby mode.

Action:

Analyze your configuration to make sure that standby capable resources can get to the *Standby* state.

- (BM, 25) Dynamic modification failed: standby capable controller `<controller>` cannot control application `<userapplication>` which has no standby capable resources on host `<sysnode>`.

In order for an application `<userapplication>` to be controlled by a controller `<controller>` the application `<userapplication>` 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) Dynamic modification failed: controller `<controller>` cannot have attributes `StandbyCapable` and `IgnoreStandbyRequest` both set to 0.

This message appears when the user sets both controller attributes `StandbyCapable` and `IgnoreStandbyRequest` to 1.

Action:

Make sure that only one is set to 1 and other to 0. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 29) 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>` must have both of its attributes `OnlineTimeout` or `StandbyTimeout` set to 0 when its attribute `Follow` is set to 1.

Action:

Contact field support.

- (BM, 42) Dynamic modification failed: application `<userapplication>` is not controlled by any controller, but has one of its attributes `ControlledSwitch` or `ControlledShutdown` set to 1.

The user wants the application `<userapplication>` to be controlled by a controller, but either or both of the application's `ControlledSwitch` or `ControlledShutdown` attributes are set to 1.

Action:

Adjust the attributes accordingly and try again. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 46) Dynamic modification failed: cannot modify a global attribute *<attribute>* locally on host *<hostname>*.

The attribute *<attribute>* is global, and cannot be modified on the local host *<hostname>*. Examples of global attributes are DetectorStartScript, NullDetector, or NonCritical.

Action:

Modify *<attribute>* globally, or modify a different attribute locally.

- (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.

The 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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 59) Error *errornumber* while reading line *<linenumber>* of .dob file -- *<errortext>*.

During dynamic modification, the base monitor attempted to read its configuration from a .dob file, but an error occurred at line *<linenumber>*. The specific OS error is indicated in *<errornumber>* and *<errortext>*.

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 = *<errornumber>*, reason: *<errortext>*.

While obtaining message queue parameters, sysdef was not able to communicate them back to the base monitor. The values of *<errornumber>* and *<errortext>* indicate the kind of error.

Action:

Contact field support.

- (BM, 71) 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"` `StandbyTransitions="No"` `AutoStartUp=0` `ControlledSwitch = 1` `ControlledShutdown = 1` `PartialCluster = 0`. However, the real values are `IndependentSwitch = <isw>` `AutoSwitchOver = <asw>` `StandbyTransitions = <str>` `AutoStartUp = <asu>` `ControlledSwitch = <csw>` `ControlledShutdown = <css>` `PartialCluster = <pcl>`.

When the controller's `Follow` attribute is set, other attributes must be set as follows: `IndependentSwitchover=0`, `AutoSwitchOver=No`, `StandbyTransitions=No`, `AutoStartUp=0`, `ControlledSwitch=1`, `ControlledShutdown=1` and `PartialCluster=0`. However, one or more of these conditions are violated in the configuration file.

Action:

Supply a valid combination of attributes for the controller and its controlled user application. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 72) 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 parent application of the controller.

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 *errornumber* - "*errortext*".

This is a generic message indicating that the execution of the routine *<routine>* failed due to the reason *<errortext>* 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` is used in new configurations as support for `hvgdstartup` may be discontinued in future releases.

- (BM, 75) 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) 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 have their AutoSwitchOver attributes set to include the option Shutdown as well.

Action:

Provide correct settings for the AutoSwitchOver attributes.

- (BM, 81) Dynamic modification failed: local controller attributes such as NullDetector or MonitorOnly cannot be modified during local modification (hvmmod -l).

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) 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 the object name is greater than the maximum allowable length.

Action:

Ensure that the length of the object name is smaller than *<maxlength>*.

- (BM, 92) 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) Dynamic modification failed: the ApplicationSequence 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 that does not appear in the its Resource list controlled applications.

Action:

Provide correct settings for ApplicationSequence and Resource attributes of the controller.

- (BM, 96) 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) Dynamic modification failed: controller <controller> attribute <ApplicationSequence> is set to <applicationsequence> which refers to application(s) not present in the configuration.

A scalable controller must list only existing applications in its ApplicationSequence attribute.

Action:

Provide correct settings for the attribute ApplicationSequence.

- (BM, 98) Dynamic modification failed: two scalable controllers <controller1> and <controller2> control the same application <application>.

Only one scalable controller can control an application.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 99) 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.

Host list mismatch between controlled and controlling applications. The controlling application must be able to run on all the hosts where the controlled applications will run.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 101) 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.

Host list mismatch between controlled and controlling applications. The controlling application must be able to run on all the hosts where the controlled applications will run.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 103) Dynamic modification failed: Controller `<controller>` has its attribute `Follow` set to 1 and the controlled application `<application>` has `StandbyCapable` resources. Therefore the controller itself must have `StandbyCapable` set to 1 and `IgnoreStandbyRequest` must be set to 0.

When a controller object's `Follow` attribute is set and the controlled application has `StandbyCapable` resources, the controller must have `StandbyCapable` set and `IgnoreStandbyRequest` must be disabled. Otherwise, standby requests will not properly be propagated to the controlled application.

Action:

Supply a valid combination of attributes for the controller and its controlled application.

- (BM, 105) 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 the 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 the `rKind` attribute and flag `-k` of the detector startup line do not match.

Action:

Fix the RMS configuration.

- (BM, 107) Illegal different values for `rKind` attribute in object *<object>*.

Different values for the `rKind` attribute were detected within the same object.

Action:

Fix the RMS configuration.

- (BM, 108) Dynamic modification failed: Scalable controller *<object>* cannot have its attribute `<SplitRequest>` set to 1.

The controller attributes `Scalable` and `SplitRequest` are mutually exclusive.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 109) 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>`.

A cluster exclusive resource cannot belong to an application that has its attribute `PartialCluster` set to 1. Likewise, a cluster exclusive resource be controlled, directly or indirectly, by a Follow controller from an application that has its attribute `PartialCluster` set to 1.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 110) 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 while its `AutoSwitchOver` setting includes the `ShutDown` option.

Action:

Correct the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 111) Dynamic modification failed: Line `#line` is too big.
A line in a configuration file is too long.

Action:

Fix RMS configuration so that every line is shorter than 2000 bytes.

- (BM, 113) Base monitor has reported 'Faulted' for host `<hostname>`.

The base monitor has reported a faulted state for the specified host and will issue node a node elimination request.

Action:

Determine the reason for the faulted state and correct it.

- (BM, 114) Dynamic modification failed: The SatNode *<satnode>* specified is already an existing SysNode entry.

The SatNode *<satnode>* is a duplicate of an existing SysNode.

Action:

Change the name of the SatNode and retry the modification.

5.5 CML: Command line

- (CML, 11) Option (*option*) requires an operand.

Certain options for `hvcml` require an argument. If `hvcml` has been invoked without the argument, this message appears along with the usage and RMS exits with exit code 3.

Action:

Check the `hvcml` man page for correct usage.

- (CML, 12) Unrecognized option *option*.

The option provided is not a valid one.

Action:

Check the `hvcml` 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 `hvcml` 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 `hvcml` is greater than the maximum possible loglevel *<maxloglevel>*, RMS exits with exit code 4.

Action:

Specify a loglevel between 1 and *<maxloglevel>* for `'hvcml -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 `hvcml`, 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 `hvcml` is not a number, 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 `hvcml` is outside the valid range, RMS exits with exit code 4.

Action:

The valid range for the `-l` option of `hvcml` is `1..<maxloglevel>`.

5.6 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, RMS generates this error.

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 that ensure the hosts are synchronized with respect to a particular operation. The messages may be transmitted 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, RMS generates this error.

Action:

Contact field support.

- (CRT, 3) Message send failed.

When RMS tries to send a message to another host in the cluster, and the delivery of this message over the queue NET_SEND_Q has failed, RMS generates this error. This may be because the host that is to receive the message has gone down, or because 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, 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.

5.7 CTL: Controllers

- (CTL, 1) Controller *<controller>* will not operate properly since its controlled resource *<resource>* is not in the configuration.

A resource is not in the configuration that is controlled by a controller, and the controller's `NullDetector` attribute is set to 0.

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.

5.8 CUP: userApplication contracts

- (CUP, 2) *userapplication*: cluster is in inconsistent condition current online host conflict, received: *hostname*, local: *onlinenode*.

The cluster hosts were unable to determine which host is responsible for *<userapplication>*. The most likely reason for this is an erroneous operator intervention (e.g., a forced `hvs` switch request) that left the `userApplication` online on more than one host simultaneously.

Action:

Analyze the cluster inconsistency and perform the appropriate action to correct it. If the application is online on more than one host, shut down (`hvutil -f`) the `userApplication` on all but one host.

- (CUP, 3) *userapplication* is already waiting for an event cannot set timer!

Critical internal error.

Action:

Contact field support.

- (CUP, 5) *userapplication* received unknown contract.

The contract received by the node from the application is not recognizable. Critical internal error.

Action:

Contact field support.

- (CUP, 7) *userapplication* is locally online, but is also online on another host.

The application is online on the local host, and it is also online on another host.

Action:

An application can only be online on one host. Make sure the application is offline on all but one of the hosts. If necessary, 'hvutil -f' to take the application offline on the superfluous hosts.

- (CUP, 8) *userapplication*: could not get an agreement about the current online host; cluster may be in an inconsistent condition!

The cluster hosts were unable to determine which host is responsible for a particular *userApplication*. The most likely reason for this is an erroneous operator intervention (e.g., an 'hvswitch -f' request) that left the *userApplication* online on more than one host simultaneously.

Note: This message corresponds to (CUP, 2). While (CUP, 8) appears in the switchlog on the contract originator, (CUP, 2) appears in the switchlog on the non-originator hosts.

Action:

Analyze the cluster inconsistency and perform the appropriate action to correct it. If the application is online on more than one host, shut it down on all but one host using 'hvutil -f'.

5.9 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.

A detector unexpectedly reported the 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: *errortext*.

The detector <detector> could not be started due to <errortext>, which 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 its 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 and terminates the restart cycle.

Action:

Contact field support.

5.10 GEN: Generic detector

- (GEN, 1) Usage: *command* -t time_interval -k kind [-d]
<command> has been invoked in a way that does not conform to its expected usage.

Action:

Use the specified syntax for the command.

- (GEN, 2) Memory lock failed.

Critical internal error.

Action:

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, 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.

5.11 INI: init script

- (INI, 1) Cannot open file *dumpfile*, `errno = errornumber: errortext`.

This message appears when the file `<dumpfile>` failed to open because of the error code `<errornumber>`, explained in `<errortext>`.

Action:

Correct the problem according to `<errortext>`.

- (INI, 9) Cannot close file *dumpfile*, `errno = errornumber: errortext`.

This message appears when the file `<dumpfile>` failed to close because of the error code `<errornumber>`, explained in `<errortext>`.

Action:

Correct the problem according to *<errortext>*.

5.12 MIS: Miscellaneous

- (MIS, 1) No space for object.

Critical internal error.

Action:

Contact field support.

5.13 QUE: Message queues

- (QUE, 13) RCP fail: *filename* is being copied.

An attempt was made to copy the file *<filename>* when there was another copy in progress.

Action:

Make sure that concurrent copies of the same file do not occur.

- (QUE, 14) RCP fail: fwrite errno *errornumber*.

There was a problem while transferring files from one cluster node to another.

Action:

Take action based on the *<errornumber>*.

5.14 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>`: *errortext*.

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 *hostname* has failed: *errortext*.

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 the appropriate `'hvutil -{o, u}'` command as needed.

- (SCR, 21) Failed to execute the script `<script>`, `errno = <errornumber>`, error reason: *errortext*.

The `<script>` could not be executed.

Action:

Take action based on the *errortext*.

- (SCR, 26) The `sdtool` notification script has failed with status *status* after dynamic modification.

After dynamic modification, the Shutdown Facility is notified via `sdtool` about the changes in the current configuration. In this case, the `sdtool` notification script failed.

Action:

Verify that `sdtool` and the Shutdown Facility are operating properly.

5.15 SWT: Switch requests (hvswitch command)

- (SWT, 4) *object* is online locally, but is also online on *onlinenode*.

The object *<object>* is online on more than one host.

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. The next host in the priority list of that particular *userApplication* tried to take over the switch request, but another host prevented the operation. 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 *<userapplication>*.

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.

5.16 SYS: SysNode objects

- (SYS, 1) Error on SysNode: *object*. It failed to send the kill success message to the cluster host: *hostname*.

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 *hostname* via a 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 the "RMS" suffix 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 that is large enough to allow a shutdown on all hosts. Check if shutdown failed due to internal problems (e.g., a failure of an `OfflineScript` prevented a userApplication from going 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 *hostname* 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 a `'hvutil -o <sysnode>'` command to clear the Wait state of the SysNode, but the SysNode is still in the Wait state because the last detector report for the cluster host `<hostname>` is not Online. That is, the SysNode might have transitioned to the Wait state not from the Online state 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`, RMS generates this error.

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, RMS generates this error.

Action:

Check if there is a problem with the network.

- (SYS, 100) The value of the attribute `<attr>` specified for SysNode `<sysnode>` is `<invalidvalue>` which is invalid. Ensure that the entry for `<attr>` is resolvable to a valid address.

The value of `<attr>` is not resolvable to a valid network address.

Action:

Ensure that a valid interface is specified for `<attr>`.

5.17 UAP: userApplication objects

- (UAP, 1) Request to go online will not be granted for application `<userapplication>` since the host `<sysnode>` runs a different RMS configuration.

This message appears when the request is done for an application `<userapplication>` 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`.
The priority list `<list>` has 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, RMS generates this error. 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 that 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, 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, RMS generates this error. 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.

A userApplication object found a task in the list of outstanding contracts but was unable to process it due to an unrecognized type of contract request. 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!

The state of the application is Offline or Standby and some of its resources are Online while others are 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 *<hostname>* into local priority list.

Internal error.

Action:

Critical error. Contact field support.

- (UAP, 44) *<object>*: could not remove *<hostname>* from local priority list.

Internal error.

Action:

Critical error. Contact field support.

- (UAP, 45) *<object>*: could not remove *<hostname>* from priority list.

Internal error.

Action:

Critical error. Contact field support.

- (UAP, 51) Failed to execute the `fcntl` system call to *flags* the file descriptor flags for file *filename*: `errno = <errornumber>`: *<errortext>*.

RMS is unable to execute the `fcntl()` system call to *<flags>* the file descriptor flags of file *<filename>* because of error code *<errornumber>* as explained by *<errortext>*.

Action:

Contact field support.

5.18 US: us files

- (US, 5) The cluster host *hostname* is no longer reachable! Please check the status of the host and the ethernet connection.

The local cluster host detected that another cluster host *<hostname>* was no longer reachable. In other words, this cluster host sees the other host *<hostname>* as faulted. The other host *<hostname>* may have gone down, or there may some problem with the cluster interconnect.

Action:

See if the host *<hostname>* is indeed dead. If not, see if there is a problem with the network 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 attempts to kill the host *<hostname>*.

Action:

Check the syslog on the host *<hostname>* to determine the reason why it went down.

- (US, 31) FAULT REASON: Resource *resource* transitioned to a Faulted state due to a detector report.

A detector unexpectedly reported the Faulted state.

Action:

Check to see if there is any problem with *<resource>*.

5.19 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 a controller propagates a request to its controlled applications, it waits for a period of time sufficient for the controlled applications to process and complete the request. When the request is not completed within this period, the controller faults.

Action:

Fix the scripts of the controller and/or the controlled applications, or repair the resources of the controlled applications. For user defined controller scripts, increase their `ScriptTimeout` values.

- (WLT, 9) `sdtool` notification timed out after *<timeout>* seconds.

After dynamic modification, 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 will issue this message and remain running.

Action:

Verify that `sdtool` and the Shutdown Facility are operating properly. Increase the `ScriptTimeout` value if needed.

5.20 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.

Internal error.

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 *<errornumber>*, 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 *<errornumber>* 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 *<errornumber>*, 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 *<errornumber>* 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.
Internal error.
Action:
Critical error. Contact field support.
- (WRP, 7) getservbyname failed for service name: *servicename*.
Internal error.
Action:
Critical error. Contact field support.
- (WRP, 8) gethostbyname failed for remote host: *hostname*.
Internal error.
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.
Internal error.
Action:
Critical error. Contact field support.
- (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, 13) Cannot allocate memory, errno *<errornumber>* - *errortext*.

The attempts to allocate memory have failed. *<errornumber>* and *<errortext>* indicate the reason.

Action:

Ensure the cluster node has sufficient memory and restart RMS.

- (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, RMS generates this error.

Action:

Contact field support.

- (WRP, 15) *gethostbyname(hostname)*: host name should be in */etc/hosts*

The host *<hostname>* specified as a *SysNode* does not have an entry in */etc/hosts*.

Action:

Add an entry for *<hostname>* to */etc/hosts*.

- (WRP, 16) No available slot for host *hostname*

RMS had already filled all of its 64 cluster interface slots when it encountered a request to allocate additional slots for host *<_HOSTNAME>*.

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>`

Internal error.

Action:

Critical error. Contact field support.

- (WRP, 23) The child process `<cmd>` with pid `<pid>` could not be killed due to errno `<errornumber>`, reason: `errortext`.

The child process with pid `<pid>` could not be killed due to the reason in `<errortext>`.

Action:

Take action based on `<errortext>`.

- (WRP, 24) Unknown flag option set for 'killChild'.

The `killChild` routine accepts only the `KILL_CHILD` and `DONTKILL_CHILD` flags, but the specified flag is neither of these.

Action:

Contact field support.

- (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:

Contact field support.

- (WRP, 29) RMS on the local host has received a message from host `hostname`, 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 `<hostname>` whose address is not resolvable by the local host.

Action:

Make sure that the local host is able to resolve the remote host `<hostname>`'s address by checking for any misconfigurations.

- (WRP, 30) RMS on the local host has received a message from host *hostname*, 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 *<hostname>* whose address is not resolvable by the local host.

Action:

Check for misconfigurations that would prevent the local host from resolving the address of remote host *<hostname>*.

- (WRP, 31) RMS has received a message from host *hostname* 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 *<hostname>* with IP address *<receivedip>*, but that address is different from the locally calculated IP address *<calcip>* for that host.

Action:

Check */etc/hosts* for any misconfiguration.

- (WRP, 32) RMS has received a message from host *hostname* 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 *<hostname>* 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 until the number of received messages reaches 500. After that, this message will be printed for every 250 such messages received.

Action:

Check */etc/hosts* for any misconfiguration.

- (WRP, 33) Error while creating a message queue with the key *<id>*, errno = *<errornumber>*, explanation: *<errortext>*.

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 *hostname* 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 *<hostname>* is not in sync with the time on the local node.

Action:

Sync the time on *<hostname>* with the time on the local node.

- (WRP, 35) Cluster host *hostname* is no longer in time sync with local node. Sane operation of RMS can no longer be guaranteed.

The time on the cluster host *<hostname>* differs significantly (> 25 seconds) from the local node.

Action:

Make sure that all the cluster hosts are in time sync.

- (WRP, 52) The operation *func* failed with error code *errornumber*.

The operation *<func>* failed with error code *<errornumber>*.

Action:

Contact field support.

- (WRP, 60) The elm heartbeat detects that the cluster host *<hostname>* has become offline.

The ELM status has failed.

Action:

Check if `bm` on the remote is alive.

6 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
- WRP: Wrappers

6.1 ADC: Admin configuration

- (ADC, 16) Because some of the global environmental variables were not set up in `hvenv` file, RMS cannot startup. 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` and `HV_RCSTART` have to be set in `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`. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (ADC, 21) Because some of the local environmental variables were not set up in `hvenv` file, RMS cannot startup. Shutting down.

Some of the local environment variables were set in the `hvenv` file. RMS 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. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (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.

- (ADC, 73) The environment variable `<hvenv>` has value `<value>` which is out of range.

The value of environment variable `<hvenv>` is out of range. The base monitor will exit.

Action:

Use a valid value for the environment variable and restart RMS.

6.2 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, 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. During this phase, if it encounters errors in its configuration file, RMS exits with exit code 23.

Action:

Check the previous messages in the switchlog to see which errors RMS detected in the configuration file.

6.3 BM: Base monitor

- (BM, 3) Usage: *progname* [-c config_file] [-m] [-h time] [-l level] [-n]

An attempt has been made to start RMS in a way that does not conform to its expected usage. RMS exits with exit code 3.

Action:

Start RMS with the correct syntax.

- (BM, 49) Failure calculating configuration checksum.
During dynamic reconfiguration, RMS calculates the configuration checksum by using `/usr/bin/sum`. If this fails, RMS exits with 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 *errornumber* - "*errortext*"

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 `<errortext>` gives the reason for the error.

Action:

Contact field support.

- (BM, 58) Not enough memory -- RMS cannot continue its operations and is shutting down.

RMS is shutting down because it does not have enough memory 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 the configuration's checksum. Therefore, it shuts down.

Action:

The previous message in the switchlog explains why RMS has not been able to write the configuration file. 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, 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>`.

- (BM, 116) The RMS-CF interface is inconsistent and will require operator intervention. The CF layer is not yet initialized.

This is a generic message indicating the RMS-CF interface is inconsistent because CF is not yet initialized.

Action:

After Cf is initialized, retry starting the base monitor.

- (BM, 117) The RMS-CIP interface state on the local node cannot be determined due to error in `popen()` -- `errno = errornumber: errortext`.

The RMS-CIP interface state cannot be determined due to failure of `popen()`, which is used internally to call `ciptool` on the local node.

Action:

Correct the `popen()` problem and then try restarting the base monitor.

- (BM, 118) The RMS-CIP interface state on the local node is required to be "UP", the current state is *state*.

The RMS-CIP interface state must be UP for proper RMS startup.

Action:

After CF is successfully initialized, try restarting the base monitor.

6.4 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.

6.5 CMM: Communication

- (CMM, 1) Error establishing outbound network communication.

If there is an error in creating outbound network communication, 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, RMS exits with exit code 12.

Action:

System error. Contact field support.

6.6 CRT: Contracts and contract jobs

- (CRT, 6) Fatal system error in RMS. RMS will shut down now. Please check the bmllog for SysNode information.

A system error has occurred within RMS.

Action:

Contact field support.

6.7 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, 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, RMS exits with exit code 12.

Action:

Contact field support.

- (DET, 16) Cannot create gdet queue of kind *gkind*.

Each of the generic detectors has a message queue that it uses to communicate with the base monitor. If there is a problem creating a queue for a detector of kind *<kind>*, RMS exits with exit code 12.

Action:

Contact field support.

- (DET, 18) Error reading `hvgdstartup` file. Error message: *errortext*.

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 exits with exit code 26.

Action:

Contact field support.

6.8 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, 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 timeout limit allocated for its execution has expired. The timeout limit is the lesser of the values defined in the environment variable `SCRIPTS_TIME_OUT` or 300.

Action:

Increase the timeout value, or correct the conditions that lead to timeout during script execution.

- (INI, 11) `InitScript` failed to startup, errno *erronumber*, reason: *errortext*.

An error occurred during startup of `InitScript`. The `errno` code `<erronumber>` and reason `<errortext>` are displayed 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.

- (INI, 17) Controller *controller* refers to an unknown userApplication *<userapplication>*
The RMS configuration contains a controller *<controller>* that refers to an unknown userApplication object.

Action:

Correct the configuration. Use only PCS to create configurations.

- (INI, 18) Configuration uses objects of type "controller" and of type "gController". These object types are mutually exclusive!

The RMS configuration contains both a controller object and a gController object. Only one of these controller types can be used in a configuration.

Action:

Correct the configuration so that only one controller type is present. Use only PCS to create configurations.

- (INI, 19) userApplication *<childapp>* is simultaneously controlled by 2 gController objects *<controller1>* and *<controller2>*. This will result in unresolvable conflicts!

The RMS configuration contains two controllers, *<controller1>* and *<controller2>*, that both control the same userApplication *<childapp>*. A userApplication can have only one parent controller.

Action:

Correct the configuration. Use only PCS to create configurations.

- (INI, 20) Incorrect configuration of the gController object *<controller>*! The attributes "Resource" and "ControllerType" are mandatory.

The RMS configuration contains a gController object with at least one required attribute undefined.

Action:

Correct the configuration. Use only PCS to create configurations.

- (INI, 21) Incorrect configuration of the gController object *<controller>*! It has the attribute Local set, but the host list for the controlled application *<childapp>* does not match the host list for the controlling application *<parentapp>*.

The RMS configuration contains a gController object *<controller>* with its Local attribute set. This requires the parent (controlling) and child (controlled) applications to be able to run on the same host. However, their respective priority lists are different.

Action:

Correct the configuration. Use only PCS to create configurations.

6.9 MIS: Miscellaneous

- (MIS, 4) The locks directory *directory* cannot be cleaned of all old locks files: at *call*, *errno* = *errornumber*, *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, RMS exits with exit code 99. The *<call>* indicates at which stage the cleanup has failed, *<errornumber>* is the OS *errno* value, and *<errortext>* is the OS supplied explanation for *errno*.

Action:

Make sure that the lock directory *<directory>* exists.

6.10 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, RMS exits with exit code 67.

Action:

Contact field support.

- (QUE, 2) Read message failed in ADMIN_Q.

The RMS base monitor was unable to extract a message of the ADMIN_Q that is used for communication between the utilities and RMS. RMS exits with exit code 3. Critical error.

Action:

Contact field support.

- (QUE, 5) Network message read failed.

If there is a problem reading a message over the network, RMS exits with exit code 3.

Action:

Critical error. Contact field support.

- (QUE, 6) Network problem occurred.
A network problem occurred 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 exits with exit code 15.
Action:
Contact field support.
- (QUE, 12) Error status in DET_REP_Q: *status*.
The RMS base monitor encountered a problem with the queue DET_REP_Q that is used by the different detectors to report their state. RMS exits with exit code 15.
Action:
Contact field support.
- (QUE, 15) Error No *errornumber* : *<errortext>* in accessing the message queue.
There was a problem using the message queue. The error number *<errornumber>* and the text in *<errortext>* indicate the type of error. Message queues are used to communicate with the base monitor.
Action:
Contact field support.

6.11 SCR: Scripts

- (SCR, 4) Failed to create a detector request queue for detector *detectormame*.
If a detector request queue could not be created for detector *<detectormame>*, 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, RMS exits 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*. If *InitScript* fails (e.g., exits with a non-zero code or gets a signal), 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, then RMS exits 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 *errornumber*.

RMS has been unable to execute a script *<script>* for the object *<objectname>*. The error number *<errornumber>* 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 *<errornumber>* and see if the cause is evident. If not, contact field support.

- (SCR, 28) Manual Mode request failed to add satellite node *node* to Manual Mode list.

Critical error.

Action:

Contact field support.

- (SCR, 29) Manual Mode request failed to remove satellite node *node* from Manual Mode list.

Critical error.

Action:

Contact field support.

6.12 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` searches the file `/etc/hosts` to get information about the host *<hostname>*, but is unable to find a valid entry for it, 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`, RMS exits with exit code 12.

Action:

Contact field support.

6.13 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.

6.14 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 occurred. See the next message for details.

A state transition error occurred in the course of RMS state transitions. Details of the error appear in the subsequent switchlog output.

Action:

Save the error description and contact field support.

6.15 WRP: Wrappers

- (WRP, 40) The length of the *type* name specified for the host *hostname* 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 no greater than the maximum length *<maxlength>*.

- (WRP, 44) Not enough slots left in the wrapper data structure to create new entries.

RMS was started up with a configuration containing more than the supported number of SysNode objects. The base monitor will exit.

Action:

Make sure that the RMS is not started on more than the supported number of SysNode objects.

- (WRP, 45) The SysNode to the CIP name mapping for *<sysnode>* has failed.

The RMS-CF-CIP mapping for *<sysnode>* has failed.

Action:

Make sure that the */etc/cip.cf* file has valid entries.

- (WRP, 46) The RMS-CF interface is inconsistent and will require operator intervention. The routine "*routine*" failed with error code *errornumber* - "*errortext*".

This is a generic message indicating that the execution of the routine *<routine>* failed due to the reason *<errortext>* and hence the RMS-CF interface is inconsistent.

Action:

Contact field support.

- (WRP, 47) The RMS-CF-CIP mapping cannot be determined for any host as the CIP configuration file `<configfilename>` cannot be opened. Please verify that all the entries in `<configfilename>` are correct and that CF and CIP are fully configured.

The CIP configuration file `<configfilename>` could not be opened for reading.

Action:

Ensure that the configuration file `<configfilename>` exists.

- (WRP, 48) The RMS-CF-CIP mapping cannot be determined for any host as the CIP configuration file `<configfilename>` has missing entries. Please verify that all the 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.

- (WRP, 54) The heartbeat mode setting of `<hbmode>` is wrong. Cannot use ELM heartbeat method on non-CF cluster.

ELM heartbeat method is not available on non CF mode cluster.

Action:

Install CF or disable ELM mode by setting `HV_USE_ELM=0`.

- (WRP, 55) The heartbeat mode setting of `<hbmode>` is wrong. The valid settings are '1' for ELM+UDP and '0' for UDP.

The `HV_USE_ELM` setting is invalid.

Action:

Set `HV_USE_ELM` to 0 or 1.

- (WRP, 58) The ELM lock resource `<resource>` for the local host is being held by another node or application.

Internal error.

Action:

Critical error. Contact field support.

- (WRP, 64) The ELM heartbeat startup failure for the cluster host `<hostname>`.

The ELM status has failed.

Action:

Correct the ELM error or use UDP mode by setting `HV_USE_ELM=0`

- (WRP, 67) The RMS-CF-CIP mapping cannot be determined for any host as the CIP configuration file `<configfilename>` has missing entries. Please verify that all the entries in `<configfilename>` are correct and that CF and CIP are fully configured.

The 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.

7 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.

7.1 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*, 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, hvmod is aborted and exits with exit code 1.

Action:

Contact field support.

- *command*: bad state: *state*.

hvassert was performed for a state *<state>* that is not among the states that can be asserted. hvassert exits with exit code 1.

Action:

Make sure that the state specified for hvassert is assertable.

- *command*: bad timeout: *timeout*.

The timeout specified for the `hvassert` command is not a number. `hvassert` 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, 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` for writing, 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`, `hvsend` 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, `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, the utility exits with exit code 3.

Action:

Contact field support.

- *command*: Must be super-user to issue this command

In order to run the command `<command>`, the user must 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 RMS is not running, the utility exits with exit code 2.

Action:

Make sure that RMS is running before invoking each utility.

- *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, RMS exits with exit code 99.

Action:

Make sure that the locks directory `<directory>` exists.

- *resource* is not in state *state*.

The `hvassert` on an object `<resource>` for a state `<state>` discovered that the resource is not in that state. `hvassert` exits with exit code 1.

Action:

None required.

- `resource` is not in state-detail state `state`.

The `hvassert` on an object `<resource>` for a state-detail `<state>` discovered that the resource is not in that state. `hvassert` exits with exit code 1.

Action:

None required.

- `timestamp`: NOTICE: User has been warned of 'hvshut -f' and has elected to proceed.

This message confirms that 'hvshut -f' has been invoked and the user has elected to proceed.

Action:

None required.

- `<command>` failed with exit code `errornumber`

When the `hvlsgclean` 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 `<errornumber>` and then the utility exits with exit code 6.

Action:

Take action based on the exit code `<errornumber>`.

- 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-establishment includes restart of RMS on the shut down host.
Do you wish to proceed? (yes = shut down RMS / no = leave RMS running).

This prompt asks for confirmation to proceed with 'hvshut -f'.

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).

The `hvreset` command requires a response before it proceeds.

Action:

Respond to the prompt.

- Can't open modification file.

When `hvmmod` is invoked with the `-c` option, it uses a temporary file. If this file cannot be opened for writing, `hvmmod` 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:

Change the target of the `hvsend` command to the indicated node.

- Command aborted.

The user has elected not to proceed with a command.

Action:

None required.

- Command timed out!

The command could not complete its task within its timeout limit.

Action:

Retry the command.

- Could not open *localfile* or could not create temporary file *filename*

During *hvrpc* processing, *<localfile>* could not be opened for reading, or the temporary file *<filename>* could not be opened for writing. *hvrpc* exits with exit code 7.

Action:

Check the permissions on *<localfile>* to make sure it is readable.

- 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 prompt appears in response to '*hvdump -E*'. The operation will proceed 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 should be shipped to RMS support.

This message indicates that the *hvdump* utility has begun collecting the information.

Action:

None required.

- Error becoming a real time process: *errortext*

The RMS base monitor runs as a real time process on Solaris, thereby giving it higher priority over other processes. If the base monitor could not start as a real time process, this message displays the reason.

Action:

Take action based on the reason.

- Error setting up real time parameters: *errortext*

There was a problem setting up the parameters for the RMS base monitor to run as a real time process.

Action:

Take action based on the reason.

- Error while starting up bm on the remote host *<targethost>*: *errortext*

An error occurred when 'hvcm -s *<targethost>*' was invoked to start RMS on a remote host.

Action:

Take action based on the reason for the problem and then reissue 'hvcm -s'.

- Error while starting up local bm: *errortext*

Error while starting up local base monitor: *<errortext>*

Action:

Take action based on the reason.

- Failed to dup a file descriptor.

RMS was unable to dup a file descriptor while setting the environment.

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.

RMS was unable to open a pipe for communication, and it exits with exit code 1.

Action:

Contact field support.

- FATAL ERROR: RMS has failed to start!

Critical internal error.

Action:

Contact field support.

- File open failed (*path*): *errortext*.

The file *<path>* that is used by the `hvassert` utility to communicate with the RMS base monitor could not be opened. `hvassert` exits with exit code 5.

Action:

Contact field support.

- File system of directory *directory* is full!

The file system used by the `hvdump` utility is full.

Action:

Clean up this file system or use `hvdump -w <altdir>` to specify an alternate directory in a file system that is not in a critical state.

- 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.

RMS was unable to fork a process, and it 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, `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 `errornumber`

'`hvutil -A <targethost>`' was invoked, but the command failed to ascertain whether or not RMS is running on `<targethost>`. The `<errornumber>` indicates a value in `/usr/include/sys/errno.h`.

Action:

Consult the system manual pages or the appendix of this manual for the explanation for error number `<errornumber>` and see if the cause is evident. If not, contact field support.

- `hvutil`: Could not determine IP address of `<targethost>`

The name of the target cluster host could not be resolved to an IP address.

Action:

Add an entry for `<targethost>` in the `/etc/hosts` file on all cluster hosts.

- `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 "rms hb"

An entry is missing in the `/etc/services` file for the RMS heartbeat.

Action:

Add an entry on all cluster hosts for rms hb 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: Processing Manual Mode request (*request*) for satellite node *node*.
 Informational message.
Action:
 None required.

- hvutil: RMS is not running on *<targethost>*
 'hvutil -A *<targethost>*' has been invoked but RMS is not running on the target host.
Action:
 None required.

- hvutil: RMS is running on *<targethost>*
 'hvutil -A *<targethost>*' has been invoked and RMS is running on the target 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 that 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 that 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: *errortext*
hvutil was unable to open the file <path> because of <errortext>.
Action:
Contact field support.
- hvutil: Valid values are: 0 to turn off logging. Positive number 1-1024 to turn on logging and "display" to show current log level.
When 'hvutil -L' is invoked with an invalid argument, it displays this message and then exits with exit code 6.
Action:
Specify a valid argument for the utility.
- Invalid delay.
The delay specified for sending a message using hvsend was a number less than zero.
Action:
Provide a valid value for the delay.
- It may take few seconds to do Debug Information collection.
The hvdump utility prints this message when it begins collecting the information for the resource graph.
Action:
None required.
- localfile *filename* does not exist or is not an ordinary file
If the <filename> argument to hvrcp does not exist or if it is not a regular file, hvrcp exits with exit code 7.

Action:

Make sure that `<filename>` exists and is an ordinary file.

- Manual Mode request (*request*) failed for satellite node *node*.
Critical error.

Action:

Contact field support.

- Manual Mode request successfully processed for satellite node *node*.
Informational message.

Action:

None required.

- Modification file name is missing on the command line,
usage: `hvmmod [-i] [-l] -f config_file.us | -E | -L | [-i] [-l] -c "modification directives"`

The `hvmmod` utility has been invoked in a way that does not conform to its expected usage. 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 length of the name of the modification file (specified as an argument through the `-f` option) or the modification directives (specified via the `-c` option) is longer than 113 characters, `hvmmod` 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 -A' and has elected to proceed.
This message confirms that 'hvshut -A' has been invoked and the user has elected to proceed.

Action:

None required.

- NOTICE: User has been warned of 'hvshut -f -a' and has elected to proceed.

This message confirms that 'hvshut -f -a' has been invoked and the user has elected to proceed.

Action:

None required.

- NOTICE: User has been warned of 'hvshut -L' and has elected to proceed.

This message confirms that 'hvshut -L' has been invoked and the user has elected to proceed.

Action:

None required.

- RELIANT_LOG_PATH is not defined

When the `hvlogclean` utility is invoked without the `-d` option, it refers to the environment variable `RELIANT_LOG_PATH` to locate the `hvloginit` script. If the value of the variable cannot be found, 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 refers to the environment variable `RELIANT_PATH` to locate the `hvloginit` script. If the value of the variable cannot be found, the utility exits with exit code 6.

Action:

Make sure that the environment variable `RELIANT_PATH` is set to the appropriate value.

- Reset of RMS has been aborted.

The `hvreset` command requires a response before it proceeds. This message confirms that the user has elected to abort the command.

Action:

None required.

- RMS environment failure: The following required variable is not defined in RMS environment:

One of the environment variables required by RMS is missing in `hvenv`.

Action:

Critical error. Check the environment definitions in the `hvenv.*` files to see if they have been corrupted.

- 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 configuration file or put the current configuration name in it that the user wants to start.

- RMS has failed to start!
`hvc`m 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 "`hvc`m" without further options to run the default configuration

This message appears when the user tries to start RMS without the `-c` option and specifying other command line options.

Action:

When using `hvc`m with the `-c` option, '`-c <configname>`' should be the last arguments on the command line. Alternatively, to use with the default configuration, enter `hvc`m without any arguments to start RMS on the local node, and '`hvc`m `-a`' to start RMS on all nodes.

- RMS has failed to start!
invalid entry in the RMS default configuration file "*configfilename*"

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 as follows:

- `<configname>`
- `'hvcm <options> -c <configname>'`.

Refer to the `hvcm` man page for valid options in the second format.

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 `hvc` 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 `hvc` to get the format.

- `rms is dead`

The `hvrpc` utility checks whether the RMS base monitor is alive every 10 seconds. If it finds that it is not alive, `hvrpc` exits with exit code 1.

Action:

Make sure RMS is running on the host.

RMS on node *node* could not be shutdown with `hvshut -A`.

- Root access required to start `hvc`

To start RMS the user must have root access.

Action:

login as root and try `hvc`.

- Sending *data* to *resource*.

If logging is turned on, this message is printed when `<data>` is being sent to the object `<resource>`.

Action:

None required.

- Shutdown of RMS has been aborted.

This message is printed when the user decides not to proceed with the 'hvshut -L' command.

Action:

None required.

- Starting Reliant Monitor Services now

This message is printed during RMS startup.

Action:

None required.

- Starting RMS on remote host *hostname* now

This message indicates that RMS is being started on the remote host <*hostname*>.

Action:

None required.

- startup aborted per user request

When RMS is started with the '-c' option and the specified configuration file is different from the entry in CONFIG.rms, RMS asks for confirmation before proceeding. If the response is "no", then this message is printed.

Action:

None required.

- The archive file is *file*

This message indicates that the `hvdump` utility has written the information to the archive file.

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 '`errornumber`' - '`errortext`'.

The execution of the command `<command>` failed trying to reset the effective uid.

Action:

Depends on the reason given in `<errortext>`.

- The command '`command`' failed to set the effective uid information with errno '`errornumber`' - '`errortext`'.

The execution of the command `<command>` failed trying to set the effective uid.

Action:

Depends on the `<errornumber>` value. See `<errortext>` for the explanation.

- The command '`command`' may not be executed on a node configured as a satellite node. Use Adaptive Services to perform the operation.

The execution of the command `<command>` on a satellite node is not supported.

Action:

Use Adaptive Services to perform the operation.

- The command '`command`' must be executed on a core node in a satellite configuration.

The execution of the command `<command>` should be performed on a core node in a satellite configuration.

Action:

Execute the command from a core node in a satellite configuration.

- 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 must be re-activated by using PCS Wizard activation command.

This message indicates the user has tried to start RMS with a configuration different from the one named in the RMS default configuration file. The base monitor is not started, so the user will need either to change the default configuration file by re-activating the configuration via the Wizard Tools `hvw` command, or to 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 by specifying the proper option argument to the `-c` option.

- The file '*filename*' could not be opened: *errortext*

While performing an `hvdump`, the file `<filename>` could not be opened because of `<errortext>`, and `hvdump` exits with exit code 8.

Action:

Take action based on the `<errortext>`.

- The length of return message from BM is illegal (*actualelength* actual *expectedlength* expected).

The `hvassert` utility received a message from the base monitor of length `<actualelength>` when it was expecting a message of length `<expectedlength>`. `hvassert` exits with exit code 5.

Action:

Contact field support.

- The system call *systemcall* could not be executed: *errortext*

While performing an `hvdump`, the `<systemcall>` could not be executed because of `<errortext>`, and `hvdump` exits with exit code 7.

Action:

Take action based on the `<errortext>`.

- The user has invoked the `hvcn` command with the `-a` flag on a host where RMS is already running, sending request to start all remaining hosts.

If `hvcn` is invoked with the `-a` flag, 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 `hvrncp`, 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 RMS on the remote host is dead.

Action:

Make sure that RMS is running on the remote host.

- Too many arguments, usage: `hvmod -E`

The `hvmod` utility does not expect any arguments when invoked with the `-E` option. If arguments are supplied, `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.

An attempt was made to assert on a number of objects that is greater than `<maximum>`.

Action:

Make sure that the number of asserted objects is no greater than the maximum.

- Unable to access directory *directory*

This message indicates that the `hvdump` utility was unable to access the directory `<directory>` to create the compressed file.

Action:

Ensure that the directory `<directory>` exists and that it has the right permissions.

- Unable to execute command: *command*

During `hvmmod`, if the command `<command>` could not be executed, `hvmmod` exits with exit code 1.

Action:
Contact field support.
- Usage: `hvassert [-h SysNode] [-q] -s resource_name resource_state | [-h SysNode] [-q] -w resource_name resource_state seconds | [-h SysNode] [-q] -d resource_name state_detail [seconds]`

The `hvassert` utility was invoked in a way that does not conform to its expected usage. `hvassert` exits with exit code 6.

Action:
Follow the usage specified above.
- Usage: `hvcn [-V] [-a] [-s targethost] [-c config_file] [-h time] [-l level]`

The `hvcn` utility has been invoked in a way that does not conform to its expected usage.

Action:
See the `hvcn` man page for correct usage.
- Usage: `hvconfig -l | -o config_file`

The `hvconfig` utility has been invoked in a way that does not conform to its expected usage. 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]`

The `hvdisp` utility has been invoked in a way that does not conform to its expected usage. The utility exits with exit code 6.

Action:
Follow the expected usage for the utility.

- Usage: `hvdump {-g | -f out_file | -t wait_time | -w working_directory}`

The `hvdump` utility has been invoked in a way that does not conform to its expected usage. The utility exits with exit code 6.

Action:

Follow the expected usage for the utility.

- Usage: `hveject -s host`

The `hveject` utility has been invoked in a way that does not conform to its expected usage. 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`

The `hvjoin` utility has been invoked in a way that does not conform to its expected usage. 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]`

The `hvlogclean` utility has been invoked in a way that does not conform to its expected usage. The utility exits with exit code 6.

Action:

Follow the expected usage for the utility.

- Usage: `hvmmod [-i] [-l] -f config_file.us | -E | -L | [-i] [-l] -c "modification directives"`

The `hvmmod` utility has been invoked in a way that does not conform to its expected usage. The utility exits with exit code 6 in any of the following situations:

- `hvmmod` is invoked without any options.
- `hvmmod` is invoked with the `-l` or `-i` options but with arguments when none are expected.

Action:

Follow the expected usage for the utility.

- Usage: `hvrpc localfile node:remotefile`

One of the following conditions occurred while invoking `hvrpc`:

- The number of arguments specified is not equal to 2.
- The second argument is not specified in the form `<node>:<remotefile>`. `hvrpc` then exits with exit code 6.

Action:

Follow the intended usage of `hvrpc` as specified above.

- Usage: `hvreset [-t timeout] userApplication`

The `hvreset` utility has been invoked in a way that does not conform to its expected usage. 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] }`

The `hvsend` utility has been invoked in a way that does not conform to its expected usage.

Action:

Follow the intended usage of the utility.

- Usage: `hvshut {-f | -L [-q] | -a | -l [-q] | -s SysNode [-q] | -A}`

The `hvshut` utility has been invoked in a way that does not conform to its expected usage. The utility exits with the exit code 6.

Action:

Follow the usage specified above.

- Usage: `hvswitch [-f] userApplication [SysNode] | -p userApplication`

The `hvswitch` utility has been invoked in a way that does not conform to its expected usage. The utility exits with exit code 6.

Action:

Follow the intended usage of the utility.

- Usage: `hvutil {-a | -d | -c | -s} userApplication | -f [-q] userApplication | {-t n | -N string } resource | -L {level | display} resource | {-o | -u} SysNode | -l {level | display} | -w | -W | -i {all | userApplication} | -r | -m {on|off|forceoff} userApplication | -M {on|off|forceoff} | {-C | -E} userApplication`

This message could appear in any one of the following situations:

- 'hvutil -u' is invoked with more than 1 argument (except -q option). 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`.

8 Warning messages

This chapter contains a detailed list of all RMS warnings 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
- CTL: Controllers
- CUP: userApplication contracts
- DET: Detectors
- 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, 19) Clearing the cluster Wait state for SysNode `<sysnode>` by faking a successful host elimination. If `<sysnode>` is actually still online, and/or if any applications are online, this 'hvutil -u' command may result in data corruption.

The 'hvutil -u `<sysnode>`' command will be executed even though the node is in the Wait state. If the node has not been manually killed beforehand, data corruption may occur.

Action:

After the Shutdown Facility fails to kill a node, the node should be killed manually.

- (ADC, 23) File `<filename>` can't be opened: `<errortext>`

A file that was to be sent to the remote host couldn't be opened.

Action:

Check the `<errortext>` and other WARNING/ERROR messages.

- (ADC, 24) File cannot be open for read.

A file that was to be sent to the remote host couldn't be read.

Action:

The message (ADC, 23) would be output too. Check the `<errortext>` of (ADC, 23) and other WARNING/ERROR messages.

- (ADC, 51) hvshut utility has timed out.

The hvshut utility could not complete its operation within the allowed timeout period.

Action:

Issue the hvshut command again. If the timeout persists, you may have to adjust an object's `ScriptTimeout` attribute to increase the offline timeout limit.

- (ADC, 65) Since RMS on this host has already encountered other Online nodes, it will remain running. However, nodes reporting incorrect checksums will NOT be brought Online.

During the first 120 seconds after the local RMS started up, it detected one or more online nodes, but then other nodes subsequently reported incorrect checksums. RMS on the local node does not shut down. Instead, it will keep running, but nodes with the incorrect checksum will not be brought online.

Action:

Make sure that configurations on all nodes are the same.

8.2 ADM: Admin, command, and detector queues

- (ADM, 61) *object* is deactivated. Switch request skipped.

A switch request cannot be performed for a userApplication in the Deact state.

Action:

Activate the userApplication and issue the switch request again.

- (ADM, 65) System *<hostname>* is currently down.

The hvswitch command was invoked for a target host that is currently down.

Action:

Start the target host and issue the switch request again, or choose another target host.

- (ADM, 69) Shutting down RMS while resource *<resource>* is not offline.

RMS is shutting down even though a resource is not offline.

Action:

In case the shutdown request fails, see if there is a failure of an OfflineScript that prevented a userApplication from going offline.

- (ADM, 80) 'hvswitch' command ignored. Target application *<userapplication>* has its ControlledSwitch attribute set to 1, so it can only be switched by its parent controller.

The target of the switch request is a controlled application that has its ControlledSwitch attribute set to 1. This application can only be switched from its parent controller, so the switch request is cancelled.

Action:

Issue a switch request to the controlling application.

- (ADM, 105) Shutdown on target node *<sysnode>* in progress. Switch request for application *<userapplication>* skipped.

The target node of the switch request is responding to an earlier shutdown request. The switch request is cancelled.

Action:

None required.

- (ADM, 110) SysNode *<node>* has been marked as going down, but failed to go offline. Check for a possibly hanging shutdown. To avoid cluster inconsistency, this SysNode cannot rejoin the cluster until it completes its shutdown.

A timeout occurred during the shutdown of SysNode *<node>*.

Action:

Check for a possibly hanging shutdown and then try to shut down the node again.

- (ADM, 111) Timeout occurred for local hvshut request. Reporting a failure back to the command now.

A timeout occurred for an RMS shutdown request.

Action:

Check for a possibly hanging shutdown and then try to shut down RMS again.

- (ADM, 113) Terminating due to a timeout of RMS shutdown. All running scripts will be killed.

A timeout occurred for a local RMS shutdown request, and script processing is not finished. All the currently running scripts will be killed.

Action:

Check for a possibly hanging shutdown and then try to shut down RMS again.

- (ADM, 114) *userapplication*: Shutdown in progress, and `AutoSwitchOver` attribute might include the `ShutDown` option, but the application failed to reach a settled `Offline` state. Switchover must be skipped.

While shutting down RMS, the `userApplication` failed to reach a settled `Offline` state. In this case, even if the `AutoSwitchOver` attribute includes the `ShutDown` option, switchover is cancelled.

Action:

Check whether RMS shutdown is completed and switchover is cancelled. If so, invoke a manual request to switch the `userApplication`. Also, check the logs to determine why the `userApplication` failed to go offline.

- (ADM, 115) Received "old style" shutdown contract, but no host with RMS 4.0 is member of the cluster. Discarding the contract.

Even though there are no nodes running RMS 4.0 and lower, this node received the old style contract. The contract will be discarded.

Action:

None required.

- (ADM, 116) Received "new style" shutdown contract, but at least one host with RMS 4.0 is member of the cluster. Discarding the contract.

The node received a new style contract when it was expecting an old style contract. The contract will be discarded.

Action:

None required.

8.3 BAS: Startup and configuration errors

- (BAS, 1) Object *<object>* is not offline.
Offline processing for the *<object>* failed. The object is still partially online, so the switch request will be cancelled.

Action:

Check the logs to determine why offline processing failed for *<object>*.

- (BAS, 8) Object *<object>* has no `rName` attribute. The `rName` attribute is normally used by the generic detector to determine which resource to monitor. Be sure that your detector can function without an `rName` attribute.

The object *<object>* does not have an `rName` attribute defined. This attribute is required by the generic RMS detectors, but may be absent for custom detectors.

Action:

None required if the corresponding custom detector is properly designed. However, if you expect to use a generic detector with this object, either now or in the future, specify an `rName` attribute.

- (BAS, 22) `DetectorStartScript` for kind *<kind>* is not defined in either `.us` or `hvgdstartup` files, therefore RMS will use default `g<kind> -k<kind> -t<timeperiod>`.

The `DetectorStartScript` for *<kind>* is not defined, so RMS will use the default detector start script.

Action:

If the default detector script is acceptable then no further action is required. Otherwise, make sure that `DetectorStartScript` is properly defined.

8.4 BM: Base monitor

- (BM, 4) The CF cluster timeout `<cftimeout>` exceeds the RMS timeout `<rmstimeout>`. This may result in RMS node elimination request before CF timeout is exceeded. Check the CF timeout specified in `/etc/default/cluster.config` and the RMS heartbeat timeout specified by `'hvcm -h'`.

The CF cluster timeout exceeds the RMS timeout. This may result in RMS node elimination requests before the CF timeout is exceeded.

Action:

Check the CF timeout specified in `/etc/default/cluster.config` and the RMS heartbeat timeout specified by `'hvcm -h'`.

- (BM, 8) Failure sending message `<message>` to object `<object>` on host `<hostname>`.

RMS failed to send a message to a remote host.

Action:

None required. Message transmission is always retried in the event of a failure.

- (BM, 28) Application `<userapplication>` has `ControlledSwitch` attribute set to 1, so it can only be switched on or off from the controller. `'hvutil -f/-c'` command ignored.

An `'hvutil -f'` or `'hvutil -c'` command specified a controlled application, but the application's `ControlledSwitch` attribute is set.

Therefore, only the parent controller can perform these operations.

Action:

Specify the controlling application on the `'hvutil -f'` or `'hvutil -c'` command.

- (BM, 30) 85. Ignoring dynamic modification failure for object `<object>`: attribute `<attribute>` is invalid.

The attribute `<attribute>` is not valid for object `<object>`.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 31) 86. Ignoring dynamic modification failure at line `<linenumber>`: cannot modify attribute `<attribute>` of object `<object>` with value `<value>` because the attribute does not exist.

The attribute `<attribute>` specified at line `<linenumber>` of the configuration file is not valid for object `<object>`.

Action:

Fix the RMS configuration. Use only standard PRIMECLUSTER configuration tools to create and maintain configurations.

- (BM, 53) The RMS-CF-CIP mapping cannot be determined for any host because the CIP configuration file `<configfilename>` cannot be opened. Verify that all entries in `<configfilename>` are correct and that CF and CIP are fully configured.

The RMS-CF-CIP mapping cannot be determined for any host because the CIP configuration file cannot be opened.

Action:

Verify that all entries in the CIP configuration file are correct and that CF and CIP are fully configured.

- (BM, 70) Some messages were not sent during RMS shutdown.

Some messages were not sent to local or remote RMS processes while RMS was shutting down.

Action:

Make sure that OS message queue parameters and network conditions allow messages to be transmitted between local and remote RMS processes.

- (BM, 76) Failed to find 'rmshb' port address in `/etc/services`. The `'hvutil -A'` command will fail until a port entry for 'rmshb' is made in the `/etc/services` file and RMS is restarted.

The `rmshb` port entry in `/etc/services` does not exist.

Action:

Add the entry for `rmshb` in `/etc/services` and restart RMS.

- (BM, 77) Failed to allocate a socket for 'rms hb' port monitoring.

The `socket()` call failed to allocate a port for rms hb.

Action:

Contact field support to help determine why there are no sockets available.

- (BM, 78) The reserved port for 'rms hb' appears to be in use. The 'rms hb' port is reserved in the `/etc/services` file but another process has it bound already. Select another port by editing the `/etc/services` file, propagating this change to all nodes in the cluster, and then restarting RMS.

RMS could not `bind()` the rms hb network port.

Action:

Edit the `/etc/services` file, select a new port number for rms hb, and then restart RMS. This port number must be identical on all cluster nodes.

- (BM, 82) A message to host `<remotehost>` failed to reach that host after `<count>` delivery attempts. Communication with that host has been broken.

A communication breakdown prevented delivery of a message between the local and remote RMS monitors.

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.

- (BM, 83) Failed to execute the `fcntl` system call.

RMS was unable to set the close-on-exec flag using `fcntl`.

Action:

Contact field support.

- (BM, 85) Application `<userapplication>` has its `ControlledSwitch` attribute set to 1, so it can only be deactivated by the parent controller. `'hvutil -d'` command ignored.

The command `'hvutil -d <userapplication>'` was invoked, but the specified target is a controlled application that its `ControlledSwitch` attribute set to 1. Therefore, it can only be deactivated by the parent controller.

Action:

Specify the controlling application on the `'hvutil -d'` command.

- (BM, 86) Application `<userapplication>` has its `ControlledSwitch` attribute set to 1, so it can only be deactivated by the parent controller. `'hvutil -D'` command ignored.

The command `'hvutil -D <userapplication>'` was invoked, but the specified target is a controlled application that its `ControlledSwitch` attribute set to 1. Therefore, it can only be deactivated by the parent controller.

Action:

Specify the controlling application on the `'hvutil -D'` command.

- (BM, 112) Controller `<controller>` has its `Follow` attribute set to 1, while its `ClusterExclusive` attribute is set to 0. However, it is controlling, directly or indirectly via a chain of `Follow` controllers, an application `<application>`. That application contains a resource named `<resource>` that has its `ClusterExclusive` attribute set to 1. This is not allowed because the application, along with its child resource, could potentially go online on more than one host. Cluster exclusive resources must be controlled by cluster exclusive `Follow` controllers.

A `Follow` controller that is not cluster exclusive contains a cluster exclusive resource in its graph.

Action:

Fix the RMS configuration so that the `Follow` controller is also cluster exclusive. Alternatively, depending on the nature of the resource, it may be possible to remove its cluster exclusive restriction.

- (BM, 119) The RMS base monitor failed to be locked in memory via `mlockall()`. Reason: `<errortext>`.

When the environment variable `HV_MLOCKALL` is set to 1, the base monitor process and any memory it allocates will be locked in memory. In this case, the RMS base monitor could not be locked in memory, but it will continue to run using unlocked memory.

Action:

Check `<errortext>` for the reason, and make sure there is enough memory.

8.5 CTL: Controllers

- (CTL, 6) Controller `<controller>` has detected more than one controlled application Online.

If the controller has two or more of the controlled applications online on one or more hosts, then the controller faults.

Action:

Make sure that no more than one controlled application for a controller is online.

- (CTL, 7) Controller `<controller>` has its attribute `IgnoreOnlineRequest` set to 1 and its `OnlineScript` is empty. Therefore, an online request to the controller might fail to bring the controlled application online.

The controller has its `IgnoreOnlineRequest` attribute set to 1 and its `OnlineScript` is empty. Therefore, an online request to the controller might fail to bring the controlled application online.

Action:

If the behavior is acceptable, then no further action is required. Otherwise, if the controller needs to bring the controlled application online, then consider changing the settings.

- (CTL, 8) Controller *<controller>* has its attribute `IgnoreOfflineRequest` set to 1 and its `OfflineScript` is empty. Therefore, an offline request to the controller might fail to bring the controlled application offline.

The controller has its `IgnoreOfflineRequest` attribute set to 1 and its `OfflineScript` is empty. Therefore, an offline request to the controller might fail to bring the controlled application offline.

Action:

If the behavior is acceptable, then no action is required. Otherwise, consider changing the settings.

- (CTL, 11) Controller *<controller>* has its attribute `StandbyCapable` set to 1, its attribute `IgnoreStandbyRequest` set to 1, and its `OnlineScript` is empty. Therefore, a standby request to the controller might fail to send the controlled application to the Standby state.

The controller has its `StandbyCapable` attribute set to 1, its `IgnoreStandbyRequest` attribute set to 1, and its `OnlineScript` is empty. Therefore, a standby request to the controller might fail to send the controlled application to the Standby state.

Action:

If the behavior is acceptable, then no action is required. Otherwise, consider changing the settings.

8.6 CUP: userApplication contracts

- (CUP, 1) *userapplication*: priority list conflict detected. Trying again...

A priority list conflict was detected. RMS automatically retries the operation.

Action:

None required.

- (CUP, 9) *userapplication*: Processing of current online host contract is not yet settled. Switch Request skipped.
A switch request was cancelled because processing of a current online host contract is not yet settled.
Action:
If the userApplication didn't go online, invoke a manual switch request.
- (CUP, 11) *userapplication*: Offline processing failed, and the application is still partially online. Switch request skipped.
The offline processing for the userApplication failed, and the userApplication is still partially online, so the switch request is cancelled.
Action:
Check the log files to see why the offline processing failed.
- (CUP, 12) *userapplication*: The required target node is not ready to go online. Switch request skipped.
A switch request specified a target node that was not yet online.
Action:
After the target node goes online, issue the request again.
- (CUP, 13) *userapplication*: No available node is ready to go online. Switch request skipped.
A switch request was invoked but there is no online node.
Action:
After a node goes online, issue the request again.
- (CUP, 14) *userapplication* did not get a response from <sender>.
A timeout occurred during the contract processing.
Action:
If the userApplication didn't eventually go online, make sure that the userApplication is not online on any of the other nodes, and then invoke a manual switch request.

- (CUP, 15) *userapplication*: Target host `<hostname>` is no longer available.

A switch request was invoked, but the target host is not available.

Action:

None required.

- (CUP, 16) *userapplication*: Offline processing failed, and the application is still partially online. Switch request skipped.

The offline processing for the `userApplication` failed and the `userApplication` is still partially online, so the switch request is cancelled.

Action:

Check the log files to see why the offline processing failed.

- (CUP, 17) *userapplication*: current online host request for host `<hostname>` accepted. Local inconsistency has been overridden with the forced switch option (`'hvswitch -f'`).

Although a local `Inconsistent` state existed, the current online host request with the forced switch option (`'hvswitch -f'`) has been accepted. The local inconsistency has been overridden.

Action:

None required.

- (CUP, 18) *userapplication*: current online request of host `<hostname>` denied due to a local inconsistent state.

The current online host request is denied due to a local `Inconsistent` state.

Action:

You can either clear the `Inconsistent` state first, or you can override this restriction by using the forced switch option (`'hvswitch -f'`).

- (CUP, 19) *userapplication*: is online locally, but is inconsistent on another host. Trying to force a CurrentOnlineHost contract.

The application is currently online on the local host but is inconsistent on another host. The application is switched to another host with the forced switch option to override the inconsistency.

Action:

None required.

- (CUP, 20) *userapplication*: AutoStartUp skipped. Application is inconsistent on host <hostname>.

The AutoStartUp processing is cancelled due to the Inconsistent state.

Action:

Clear the Inconsistent state and switch the application online.

- (CUP, 21) *userapplication*: Failover skipped. Application is inconsistent on host <hostname>.

The failover processing is cancelled due to the Inconsistent state.

Action:

Clear the Inconsistent state.

- (CUP, 22) *userapplication*: Switch request skipped. Application is inconsistent on host <hostname>.

The switch request is cancelled due to the Inconsistent state.

Action:

Clear the Inconsistent state.

- (CUP, 23) *userapplication*: Switch request skipped. Application is inconsistent on local host.

The switch request is cancelled due to the Inconsistent state.

Action:

Clear the Inconsistent state.

- (CUP, 24) *userapplication*: Switch Request processed. Local inconsistency has been overridden with the forced switch option.

Although a state is inconsistent, a switch request with the forced switch option ('hvswitch -f') is accepted and the local inconsistency has been overridden.

Action:

None required.

- (CUP, 25) *userapplication* is currently in an inconsistent state. Switch request skipped. Clear inconsistency first, or override with the forced switch option.

The userApplication is currently in an Inconsistent state on the local host. The application cannot be switched until the inconsistency is resolved, so the switch request is cancelled.

Action:

You can either clear the inconsistency first, or you can override this restriction by using the forced switch option ('hvswitch -f').

- (CUP, 26) *userapplication*: LastOnlineHost conflict detected. Processing an AutoStart or PrioritySwitch CurrentOnlineHost Contract with OnlinePriority enabled. Target host of switch request is host <hostname>, but the local host is the LastOnlineHost. Denying the request.

A LastOnlineHost conflict is detected and the local host is the LastOnlineHost, so the application will be brought online on the local host.

Action:

None required.

- (CUP, 27) *userapplication*: LastOnlineHost conflict occurred. Skipping local Online request, because host <hostname> has a conflicting LastOnlineHost entry.

A LastOnlineHost conflict is detected and the local host is not the LastOnlineHost, so the application will be brought online on the other host.

Action:

None required.

- (CUP, 28) *userapplication*: Cannot get deterministic information about the LastOnlineHost. Tried to switch to `<hostname>`, but `<loh>` claims to be the LastOnlineHost. Priority switch skipped. Conflict may be resolved by operator intervention: use `hvswitch` command with an explicit target host.

A LastOnlineHost conflict is detected, and RMS cannot determine the LastOnlineHost, so the application will not go online anywhere.

Action:

Invoke a switch request specifying the target host.

- (CUP, 29) *userapplication*: LastOnlineHost conflict occurred. Timestamps of conflicting LastOnlineHosts entries do not allow a safe decision, because their difference is lower than `<time>` seconds. Conflict must be resolved by operator intervention: invalidate LastOnlineHost entry via `'hvutil -i <userapplication>'`, and then invoke an explicit `hvswitch` command.

A LastOnlineHost conflict is detected, and the timestamps of conflicting LastOnlineHost entries do not allow a safe decision because their difference is lower than HV_LOH_INTERVAL. Therefore, the application will not go online anywhere.

Action:

Invalidate the LastOnlineHost entry with `'hvutil -i <userapplication>'`, and then invoke a switch request specifying the target host.

- (CUP, 30) *userapplication*: Denying maintenance mode request. `userApplication` is busy or is in the Faulted state.

A maintenance mode request, i.e., `'hvutil -m on/off'` is denied because the `userApplication` is busy or is in the Faulted state.

Action:

Clear the Faulted state and retry the maintenance mode request.

- (CUP, 31) *userapplication*: maintenance mode request was denied by the remote SysNode <hostname> because userApplication is busy, in the Faulted state, or not ready to leave maintenance mode. See remote switchlog for details.

A maintenance mode request, i.e., 'hvutil -m on/off', is denied because the userApplication is busy, is in the Faulted state, or is not ready to leave maintenance mode.

Action:

See the remote switchlog for details.

- (CUP, 32) *userapplication*: Denying maintenance mode request. The following object(s) are not in an appropriate state for safely returning to normal operation: <resource>

A maintenance mode request ('hvutil -m on/off') is denied because the resources are not in an appropriate state for safely returning to normal operation.

Action:

Fix the states of the listed resources.

- (CUP, 33) *userapplication*: Denying maintenance mode request. The initialization of the state of the userApplication is not yet complete.

A maintenance mode request ('hvutil -m on/off') is denied because the initialization of the state of the userApplication is not yet complete.

Action:

Wait for the initialization of the state of the userApplication and retry the maintenance mode request.

- (CUP, 34) *userapplication*: LastOnlineHost conflict detected. Processing an AutoStart or PrioSwitch CurrentOnlineHost Contract with OnlinePriority enabled. Target of switch request is host <hostname>, but the local host is the LastOnlineHost. The local host takes over switch request.

A LastOnlineHost conflict is detected, and the local host is the LastOnlineHost, so the application will be brought online on the local host.

Action:

None required.

8.7 DET: Detectors

- (DET, 29) Resource *<resource>*: received detector report DetReportsOnlineWarn. The WarningScript *<warningscript>* will be run.

Informational message.

Action:

None required.

- (DET, 31) Resource *<resource>*: Received detector report DetReportsOfflineFaulted. The posted state will become OfflineFault until one of the following reports is received: DetReportsOffline, DetReportsOnline, DetReportsStandby, or DetReportsFaulted.

Informational message.

Action:

None required.

- (DET, 35) Resource *<resource>*: received detector report DetReportsOnlineWarn. The WarningScript is not defined and therefore will not be run.

Informational message.

Action:

None required.

8.8 SCR: Scripts

- (SCR, 17) Resource *<resource>*: WarningScript has failed with status *<status>*.

The WarningScript of the resource has failed.

Action:

Check the log files for additional failure information.

- (SCR, 25) Controller *<resource>*: StateChangeScript has failed with status *<status>*.

The StateChangeScript of the resource has failed.

Action:

Check the log files for additional failure information.

8.9 SWT: Switch requests (hvswitch command)

- (SWT, 1) *userapplication*: AutoStartUp attribute is set, but the HV_AUTOSTART_WAIT timeout has expired. Application autostartup skipped because not all necessary cluster hosts are online.

The AutoStartUp is cancelled because the PartialCluster attribute is set to 0 and not all necessary cluster hosts could be brought online before the HV_AUTOSTART_WAIT timeout expired.

Action:

Start RMS on all necessary cluster hosts and then start the application manually if necessary.

- (SWT, 5) *object*: AutoStartUp skipped because object is faulted.

The AutoStartUp is cancelled due to the Faulted state.

Action:

Clear the Faulted state.

- (SWT, 6) *object*: AutoStartUp skipped because a Fault occurred during initialization.

The AutoStartUp is cancelled due to the Faulted state.

Action:

Clear the Faulted state.

- (SWT, 7) *object*: AutoStartUp skipped because userApplication is deactivated.

The AutoStartUp is cancelled because the userApplication is in the Deact state.

Action:

Activate the userApplication and start the application manually.

- (SWT, 8) *object*: AutoStartUp skipped because not all necessary cluster hosts are online.

The AutoStartUp is cancelled because the PartialCluster attribute is set to 0 and not all necessary cluster hosts are online.

Action:

Start RMS on all necessary cluster hosts and then start the application manually if necessary.

- (SWT, 11) *object*: no responsible node available. Switch request skipped.

The switch request is cancelled because no responsible node is available.

Action:

Enable a responsible node and then issue the request again.

- (SWT, 12) *object* is busy or locked. Switch request skipped.

The switch request is cancelled because *<object>* is either busy or locked.

Action:

Wait until *<object>* is in a switchable state and then issue the request again.

- (SWT, 13) Not all necessary cluster hosts for application *<userapplication>* are online. Switch request skipped. If the application should be brought online anyway, use the forced

switch option ('hvswitch -f'). CAUTION: Forcing the application online could result in an inconsistent cluster if the application is already online on another node.

The switch request is cancelled because not all necessary cluster hosts for the application are online.

Action:

If the application should be brought online anyway, use the forced switch option ('hvswitch -f').

- (SWT, 14) *object* is deactivated. Switch request skipped.

The switch request is cancelled because the application has been deactivated.

Action:

Activate the application and then issue the request again.

- (SWT, 16) No target host found or target host is not ready to go online. Switch request skipped.

The target host was either not found or not ready to go online, so the switch request is cancelled.

Action:

Wait for the target host to go online or start the target host.

- (SWT, 18) *object* is not ready to go online on local host. Switch request skipped.

The switch request is cancelled because the application or the local host is in a transitional state.

Action:

Wait until both the application and the local host are online and then issue the request again.

- (SWT, 19) *object*: is not ready to go online on local host. Trying to find another host.

For a priority or 'last online host' switch, if the target host of the switch is the node where the application is faulted, then the switch request is denied and the switch request is forwarded to another host in the cluster.

Action:

None required.

- (SWT, 21) *object*: Local node has Faulted or OfflineFaulted descendants, but no other node is ready to go online. Switchover skipped.

The switch request is cancelled because the local node has Faulted or OfflineFaulted descendants and no other node is ready to go online.

Action:

Clear the Faulted state.

- (SWT, 22) *object*: local node has Faulted or OfflineFaulted descendants. Forwarding switchover request to next host: <targethost>.

The switch request is forwarded to another host because the local node has Faulted or OfflineFaulted descendants.

Action:

None required.

- (SWT, 23) *object* is busy or locked. Deact request skipped.

A Deact request cannot be processed if the target application is busy or locked.

Action:

Wait until the target is in a different state and issue the request again.

- (SWT, 24) *object* is deactivated. Switch request skipped.

A switch request cannot be processed if the target application is in the Deact state.

Action:

Activate the userApplication.

- (SWT, 28) *hostname* is unknown locally.

During the processing of a switch request, the target host name couldn't be found.

Action:

Check the state of the host.

- (SWT, 30) *object* was online on `<onlinehost>`, which is not reachable. Switch request must be skipped to ensure data integrity. This secure mechanism may be overridden with the forced switch option (`'hvswitch -f'`). CAUTION: Ensure that no further access to the data is performed by `<onlinehost>`. Otherwise the use of the `'-f'` flag may break data consistency.

The remote node is marked as a current online node but it's currently offline or unreachable. This could occur due to a previous shutdown via the forced switch option (`'hvswitch -f'`), or it could be a timing issue.

Action:

If the previous shutdown was via the forced switch option and the application should be brought online anyway, use the forced switch option again. If the previous shutdown didn't use the forced switch option, this could be a timing issue, so wait a moment and try it again.

- (SWT, 31) *object* was online on `<onlinehost>`, which is not reachable. The RMS secure mechanism has been overridden with the forced switch option (`'hvswitch -f'`). Switch request is processed.

The remote node is marked as a current online node but it's currently offline or unreachable. This could occur due to a previous shutdown via `'hvshut -f'`. The forced switch request is processed.

Action:

None required.

- (SWT, 32) *object* is currently in an inconsistent state on local host. Switch request skipped. Clear inconsistency first, or override with the forced switch option (`'hvswitch -f'`).

The application is currently in an `Inconsistent` state on the local host. The application cannot be switched until the inconsistency is resolved, so the switch request is cancelled.

Action:

You can either clear the inconsistency first, or you can override this restriction by using the forced switch option (`'hvswitch -f'`).

- (SWT, 33) *object* is not ready to go online on the local host. A local inconsistent state prevents a switch to another node. Switch request skipped.

The application is currently in an `Inconsistent` state on the local host. The application cannot be switched until the inconsistency is resolved, so the switch request is cancelled.

Action:

You can either clear the inconsistency first, or you can override this restriction by using the forced switch option (`'hvswitch -f'`).

- (SWT, 34) *object* is not ready to go online on local host. Trying to find another host.

The `userApplication` is not ready to go online on the local host, so RMS forwards the switch request to the next host in its priority list.

Action:

None required.

- (SWT, 35) *object* is not ready to go online on local host. Switch request skipped.

The `userApplication` is not ready to go online on the local host so the direct switch request is cancelled.

Action:

Wait for the `userApplication` to go online and try the switch request again.

- (SWT, 36) *sysnode* is in the `Wait` state. Switch request skipped.

The node is in the `Wait` state, so the switch request is cancelled.

Action:

Wait for the node to get out of the `Wait` state and try the switch request again.

- (SWT, 37) AutoStartUp for application `<userapplication>` is ignored because `hvmmod` was invoked with the `-i` flag.

Even though its `AutoStartUp` attribute is set to 1, the application cannot start automatically because the `hvmmod -i` command overrides this feature.

Action:

None required.

- (SWT, 58) Processing policy switch request for application `<userapplication>`. The cluster host `<sysnode>` is in the `Wait` state, so no switch request can be processed. The application will go offline now.

The state of `<userapplication>` must change due to a policy switch request, but `<sysnode>` is in the `Wait` state so no switch request can be processed except for offline processing. Therefore, the application goes offline.

Action:

Wait for the node to get out of the `Wait` state and then check the state of the application. You may have to issue a manual switch request for the application.

- (SWT, 59) Processing policy switch request for application `<userapplication>`. No cluster host is available to take over this application. The application will go offline now.

The state of `<userapplication>` must change due to a policy switch request, but no cluster host is available to take over the application. Therefore, the application goes offline.

Action:

Enable a node in the application's priority list.

- (SWT, 60) Processing policy switch request for application `<userapplication>`, which is in the `Standby` state. The application will go offline now.

During a policy switch, if an exclusive application switches to a node, then all applications in the `Standby` state must go offline because they have a lower priority. This message simply warns the user that the application is in the `Standby` state and will be going offline due to the above reason.

Action:

None required.

- (SWT, 69) `AutoStartUp` for application `<userapplication>` is ignored because the environment variable `HV_AUTOSTARTUP` is set to 0.

The application doesn't start up automatically because the environment variable `HV_AUTOSTARTUP` is set to 0, and this overrides each application's `AutoStartUp` attribute.

Action:

To allow application startup according to each application's `AutoStartUp` attribute, set the environment variable `HV_AUTOSTARTUP` to 1.

- (SWT, 72) `userapplication` received Maintenance Mode request from the controlling `userApplication`. The request is denied, because the state is either `Faulted` or `Deact` or the application is busy or locked.

The maintenance mode request from the controlling `userApplication` is denied because the state is either `Faulted` or `Deact` or the application is busy or locked.

Action:

Clear the `Faulted` or `Deact` state and try it again.

8.10 SYS: SysNode objects

- (SYS, 16) The RMS internal SysNode name `<sysnode>` is not compliant with the RMS naming convention. A non-compliant setting is possible, but this will cause all RMS commands to accept only the SysNode name, and not the Unix hostname (`uname -n`), of the cluster nodes.

The RMS internal SysNode name is not compliant with the RMS naming convention of `<sysnodename>RMS`.

Action:

Changing the RMS SysNode name to `<sysnodename>RMS` is recommended.

- (SYS, 18) The SysNode *<sysnode>* does not follow the RMS naming convention for SysNodes. To avoid seeing this message in the future, rename the SysNode to use the CF-based name of the form "*<CFname>RMS*" and restart RMS.

This message appears when the RMS internal SysNode name, *<CFname>RMS*, does not match the SysNode name specified in the configuration file.

Action:

Change the SysNode name in the configuration to the RMS internal SysNode naming convention, *<CFname>RMS*.

- (SYS, 88) No heartbeat from cluster host *<hostname>* within the last 10 seconds. This may be a temporary problem caused by high system load. RMS will react if this problem persists for *<time>* seconds more.

No heartbeat from the other node within the last 10 seconds.

Action:

Check the following items:

- LAN interconnects for connectivity problems
- the state of the other node
- high system loads

- (SYS, 99) The AlternateIp attribute specified for SysNode *<sysnode>* should not be used in CF mode. Ignoring the attribute.

Although the AlternateIp attribute is specified for *<sysnode>*, it must never be used in a cluster with CF as interconnect. Therefore, the specified AlternateIp attribute is ignored.

Action:

Unset the AlternateIp attribute.

8.11 UAP: userApplication objects

- (UAP, 2) *object* got token `<token>` from node `<node>`. TOKEN SKIPPED – Reason: `<errortext>`.

This message gives a reason for skipping a particular action. For example, the following message states that a request for Offline processing for application `userApp_1` has been denied because the application is busy:

```
(UAP, 2): WARNING: userApp_1 got token UApp_ReqOffline
from node userApp_1. TOKEN SKIPPED – Reason: object is
busy.
```

Action:

If this was a result of a request from the user, then the user should retry the request. If the request originated internally, then no further action is required.

- (UAP, 3) *object*: double fault occurred and Halt attribute is set to 1. Halt attribute will be ignored, because no other cluster host is available.

The Halt attribute will be ignored if there are no more available hosts.

Action:

Make sure that there is a sufficient number of available cluster hosts.

- (UAP, 4) *object* has gone online, but is also in the HV_AUTOSTARTUP_IGNORE list of cluster hosts to be ignored on startup. The cluster may be in an inconsistent condition.

Even though an RMS cluster node is listed in the HV_AUTOSTARTUP_IGNORE environment variable, the RMS cluster node has gone online.

Action:

Check the contents of the HV_AUTOSTARTUP_IGNORE environment variable to see if it is correct.

- (UAP, 11) *object* is not ready to go online on local node. Online processing skipped.

The userApplication is not ready to go online on the local node because it is busy or in the Faulted state.

Action:

Clear the Faulted state.

- (UAP, 12) *object*: target host of switch request `<hostname>` is no longer available. Request skipped.

The target host of the switch request is no longer available so the switch request is cancelled.

Action:

Enable the target node.

- (UAP, 14) *object* is not ready to go online on local host. Switch request skipped.

The userApplication is not ready to go online on local host so the switch request is cancelled.

Action:

None required.

- (UAP, 18) `SendUAppLockContract()`: invalid token: `<token>`.

During contract processing, the invalid token is received.

Action:

Contact field support.

- (UAP, 25) `AutoStartUp` skipped by `<object>`. Reason: not all necessary cluster hosts are online.

The userApplication didn't start up automatically because not all necessary cluster hosts are online.

Action:

Enable all necessary cluster hosts.

- (UAP, 30) *object* is not ready to go online on local host. Trying to find another host.

The userApplication is not ready to go online on local host so find another host.

Action:

None required.

- (UAP, 52) *userapplication*: double fault occurred and Halt attribute is set. Halt attribute will be ignored, because attribute AutoSwitchOver is set to *<attrvalue>*.

The Halt attribute will be ignored if the AutoSwitchOver attribute is set to *<attrvalue>*.

Action:

Modify the AutoSwitchOver attribute appropriately if you want the Halt attribute to take effect.

8.12 US: us files

- (US, 10) *object*: userApplication transitions into the Online state, even though it was previously in the Faulted state according to persistent fault info. Check for possible inconsistencies.

Even though the persistent fault info was set, the userApplication went into the Online state instead of the Faulted state.

Action:

Check to see if the same application is Inconsistent or Online on other RMS nodes.

- (US, 23) *object*: double fault occurred. Processing terminated.

Further processing for *<object>* will be stopped because of the double fault.

Action:

Check the other messages in the switchlog to determine the reason for the double fault. Clear the double fault.

- (US, 28) *object*: PreCheck failed. Switch request will be cancelled now and not be forwarded to another host, because this was a directed switch request with an explicit target host.

A PreCheckScript failed during a directed switch request, i.e., the target host of the request was explicitly specified. In this case the switch request is cancelled, so it is not forwarded to the next host in the priority list.

Action:

Invoke a new switch request specifying the next host as target host. If you want RMS to forward the request automatically, you should invoke a priority switch (`hvs` without a specified target host).

- (US, 29) *object*: PreCheck failed. Trying to find another host...

A PreCheckScript failed during a priority switch request. In this case the switch request is forwarded to the next host in the priority list.

Action:

None required.

- (US, 43) *object*: PreCheck failed. Standby request cancelled.

Execution of the PreCheckScript has failed and standby processing will be stopped.

Action:

Check to see why the PreCheckScript has failed and correct the script if necessary.

- (US, 45) *object*: PreCheck failed. The switch request will be cancelled now and not be forwarded to another host, because the AutoSwitchOver attribute did not include the ResourceFailure option.

A PreCheckScript failed and the AutoSwitchOver attribute did not include the ResourceFailure option. In this case RMS will not take automatic action in the event of a script failure. The switch request is cancelled, and it is NOT forwarded to the next host in the priority list.

Action:

Invoke a new switch request specifying the next host as the target. If you want RMS to forward the request automatically, turn on the ResourceFailure option of the AutoSwitchOver attribute.

- (US, 47) *userapplication*: Processing of Clear request resulted in a Faulted state. Resuming Maintenance Mode nevertheless. Clear the fault condition before leaving Maintenance Mode.

A Clear request ('hvutil -c') was issued for an application <*userapplication*> in maintenance mode. It failed to clear the state of the graph and resulted in a Faulted state of the application.

Action:

Check the switchlog for the origin of the failure. Fix the failure condition and re-run 'hvutil -c'. Do NOT leave maintenance mode until the fault condition has been cleared.

- (US, 55) PreCheck failed for &Local& &gController& <*controller*>: the descendant controlled &userApplication& <*userapplication*> was not ready to perform a PreCheck.

The PreCheck phase of a controlling application automatically runs the PreCheckScript of each of its local-mode child applications. The PreCheck fails if one of these child applications is in a state where the PreCheckRequest cannot be performed. This is typically due to a busy state or a faulted resource.

Action:

Re-issue the request as soon as the entire graph, including all controlled applications, is in a settled state. In case of a faulted resource, clear the appropriate fault first.

8.13 WLT: Wait list

- (WLT, 6) The script of resource <*resource*> did not terminate gracefully after receiving SIGTERM.

The script of the resource did not terminate gracefully.

Action:

See if the script timeout occurred.

8.14 WRP: Wrappers

- (WRP, 11) Message send failed, queue id *<queueid>*, process *<process>*, *<name>*, to host *<hostname>*.

RMS failed to send a message to a remote host.

Action:

None required. Message transmission is always retried in the event of a failure.

- (WRP, 41) The interconnect entry *<interconnect>* specified for SysNode *<sysnode>* has the same IP address as that of the interface *<existinginterconnect>*.

Both *<interconnect>* and *<existinginterconnect>* have the same IP address.

Action:

Make sure that the interconnect entries specified have different IP addresses.

- (WRP, 51) The 'echo' service for UDP may not have been started on the local host. Ensure that the echo service is enabled.

The echo service for UDP may not have been enabled on the local host.

Action:

Make sure that the echo service is enabled and started.

9 Notice messages

This chapter contains a detailed list of selected RMS notices that appear in the switchlog. In general, notices are self-explanatory messages that track normal operations in the cluster, so they need no further documentation. However, the notices selected for this chapter warrant additional description. In some cases, an optional action is suggested.

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
BM: Base monitor
SWT: Switch requests (hvswitch command)
SYS: SysNode objects
US: us files
WRP: Wrappers

9.1 ADC: Admin configuration

- (ADC, 22) Attempting to clear the cluster Wait state for SysNode *<sysnode>* and reinitialize the Online state.

The Shutdown Facility failed to kill *<sysnode>* and return it to the Online state. RMS tries to clear the Wait state for the specified SysNode with an implicit 'hvutil -o'.

Action:

Check the logs to see why the SF kill failed.

9.2 BM: Base monitor

- (BM, 27) Application `<userapplication>` does not transition to standby since it has one or more faulted or cluster exclusive online resources.

The application will skip standby processing because at least one of its resources is faulted or cluster exclusive online.

Action:

Check the log files for any faulted resources and clear the faults if any. Otherwise, no further action is required.

- (BM, 84) The RMS-CF-CIP mapping in `<configfilename>` for SysNode name `<sysnode>` has found the CF name to be `<cfname>` and the CIP name to be `<cipname>`, previously defined as `<olscfname>`.

RMS has mapped the SysNode name to a different CF name due to a change in the CIP database.

Action:

If the CIP database is correct, then no action is necessary. Otherwise, the database should be corrected and RMS restarted.

9.3 SWT: Switch requests (hvswitch command)

- (SWT, 48) A controller-requested switchover for the application `<app>` is attempted even though `<hostname>`, where it used to be Online, is unreachable. The RMS secure mechanism has been overridden with the forced switch option (`'hvswitch -f'`), and the switch request is processed. If that host is in the Wait state, the switchover is delayed until that host becomes Online, Offline, or Faulted.

`<app>` was previously online on host `<hostname>`, but `<hostname>` is now unreachable. Therefore, `<app>` will be switched to the local host because `'hvswitch -f'` was invoked.

Action:

None required.

- (SWT, 49) Application `<app>` will not be switched Online on host `<oldhost>` because that host is not Online. Instead, it will be switched Online on host `<newhost>`.

A priority switch (`'hvswitch -p'`) for application `<app>` was invoked, but the next priority host `<oldhost>` is not online. The application will instead be switched to `<newhost>`.

Action:

None required.

9.4 SYS: SysNode objects

- (SYS, 12) Although host `<hostname>` has reported online, it does not respond with its checksum. That host is either not reachable, or does not have the local host `<localhost>` in its configuration. Therefore, it will not be brought online.

Node `<hostname>` is reported online but has not reported its checksum. This could be because it is unreachable from the local node, or because the local node is not included in its configuration.

Action:

Check the log files and configuration to see why `<hostname>` does not respond with its checksum.

9.5 US: us files

- (US, 7) `<object>`: Transitioning into a Faulted state according to persistent fault info.

The PersistentFault attribute is set for `<object>` and the resource is set as Faulted because of previous RMS activity.

Action:

None required.

- (US, 11) Temporary heartbeat failure disappeared. Now receiving heartbeats from cluster host `<hostname>` again.

After missing some heartbeats from `<HOST>` current node has started receiving heartbeats again.

Action:

Though heartbeats are back on, it is safe to check if the systems have some communication problems or if <HOST> has some load issues and correct them.

- (US, 40) *object*: Offline processing due to hvshut finished.
Offline processing was performed due to the invocation of hvshut.

Action:

None required.

- (US, 41) The userApplication <userapplication> has gone into the Online state after Standby processing.

<userapplication> went into the Online state instead of the Standby state.

Action:

None required.

- (US, 44) *object*: Fault propagation to parent ends here. Reason is either a MonitorOnly attribute of the child reporting the Fault, or the "or" character of the current object

The fault information from the child resource will not be sent up to other parent objects because of the MonitorOnly attribute of the child resource.

Action:

None Required.

- (US, 56) The userApplication <userapplication> is already Online at RMS startup time. Invoking an Online request immediately in order to clean up possible inconsistencies in the state of the resources.

The application <userapplication> is already online at the RMS start time. This is usually due to a previous ungraceful shutdown of RMS. The application is brought online now to prevent any further inconsistencies.

Action:

None required.

9.6 WRP: Wrappers

- (WRP, 21) A message cannot be sent into a Unix message queue from the process `<pid>`, `<process>`, after `<number>` attempts in the last `<seconds>` seconds. Still trying.

In the last `<seconds>` after `<number>` attempts by the process `<pid>`, `<process>`, the message could not be placed in a Unix queue because the queues were full or busy.

Action:

Check the values of system message queue tunables such as `msgmnb`, `msgtql` and others. If necessary, increase the values and reboot.

- (WRP, 22) A message cannot be sent into a Unix message queue id `<queueid>` by the process `<pid>`, `<process>`.

An attempt to communicate between RMS processes via a Unix message queue failed.

Action:

Check the values of system message queue tunables such as `msgmnb`, `msgtql` and others. If necessary, increase the values and reboot.

- (WRP, 26) Child process `<cmd>` with pid `<pid>` has been killed because it has exceeded its timeout period.

The child process `<cmd>` has been killed by the parent process because it has exceeded its timeout period.

Action:

None required.

- (WRP, 27) Child process `<cmd>` with pid `<pid>` will not be killed though it has exceeded its timeout period.

The process `<cmd>` will not be killed though it has exceeded its timeout period.

Action:

None required.

10 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. This appendix summarizes the error numbers and their meanings for the Solaris and Linux operating systems.

10.1 Solaris error numbers

Error number	Reason for error
1	Not owner
2	No such file or directory
3	No such process
4	Interrupted system call
5	I/O error
6	No such device or address
7	Arg list too long
8	Exec format error
9	Bad file number
10	No child processes
11	Resource temporarily unavailable
12	Not enough space
13	Permission denied
14	Bad address
15	Block device required
16	Device busy
17	File exists
18	Cross-device link

Table 5: Solaris *errno* error numbers and their meanings

Error number	Reason for error
19	No such device
20	Not a directory
21	Is a directory
22	Invalid argument
23	File table overflow
24	Too many open files
25	Inappropriate ioctl for device
26	Text file busy
27	File too large
28	No space left on device
29	Illegal seek
30	Read-only file system
31	Too many links
32	Broken pipe
33	Argument out of domain
34	Result too large
35	No message of desired type
36	Identifier removed
37	Channel number out of range
38	Level 2 not synchronized
39	Level 3 halted
40	Level 3 reset
41	Link number out of range
42	Protocol driver not attached
43	No CSI structure available
44	Level 2 halted
45	Deadlock situation detected/avoided
46	No record locks available

Table 5: Solaris *errno* error numbers and their meanings

Error number	Reason for error
47	Operation canceled
48	Operation not supported
49	Disc quota exceeded
50	Bad exchange descriptor
51	Bad request descriptor
52	Message tables full
53	Anode table overflow
54	Bad request code
55	Invalid slot
56	File locking deadlock
57	Bad font file format
58	Owner of the lock died
59	Lock is not recoverable
60	Not a stream device
61	No data available
62	Timer expired
63	Out of stream resources
64	Machine is not on the network
65	Package not installed
66	Object is remote
67	Link has been severed
68	Advertise error
69	Srmount error
70	Communication error on send
71	Protocol error
72	Locked lock was unmapped
73	Facility is not active
74	Multihop attempted

Table 5: Solaris *errno* error numbers and their meanings

Error number	Reason for error
75	Error 75
76	Error 76
77	Not a data message
78	File name too long
79	Value too large for defined data type
80	Name not unique on network
81	File descriptor in bad state
82	Remote address changed
83	Can not access a needed shared library
84	Accessing a corrupted shared library
85	.lib section in a.out corrupted
86	Attempting to link in more shared libraries than system limit
87	Can not exec a shared library directly
88	Illegal byte sequence
89	Operation not applicable
90	Number of symbolic links encountered during path name traversal exceeds MAXSYMLINKS
91	Error 91
92	Error 92
93	Directory not empty
94	Too many users
95	Socket operation on non-socket
96	Destination address required
97	Message too long
98	Protocol wrong type for socket
99	Option not supported by protocol
100	Error 100
101	Error 101

Table 5: Solaris *errno* error numbers and their meanings

Error number	Reason for error
102	Error 102
103	Error 103
104	Error 104
105	Error 105
106	Error 106
107	Error 107
108	Error 108
109	Error 109
110	Error 110
111	Error 111
112	Error 112
113	Error 113
114	Error 114
115	Error 115
116	Error 116
117	Error 117
118	Error 118
119	Error 119
120	Protocol not supported
121	Socket type not supported
122	Operation not supported on transport endpoint
123	Protocol family not supported
124	Address family not supported by protocol family
125	Address already in use
126	Cannot assign requested address
127	Network is down
128	Network is unreachable
129	Network dropped connection because of reset

Table 5: Solaris *errno* error numbers and their meanings

Error number	Reason for error
130	Software caused connection abort
131	Connection reset by peer
132	No buffer space available
133	Transport endpoint is already connected
134	Transport endpoint is not connected
135	Structure needs cleaning
136	Error 136
137	Not a name file
138	Not available
139	Is a name file
140	Remote I/O error
141	Reserved for future use
142	Error 142
143	Cannot send after socket shutdown
144	Too many references: cannot splice
145	Connection timed out
146	Connection refused
147	Host is down
148	No route to host
149	Operation already in progress
150	Operation now in progress
151	Stale NFS file handle

Table 5: Solaris *errno* error numbers and their meanings

10.2 Linux error numbers

Error number	Reason for error
1	Operation not permitted
2	No such file or directory
3	No such process
4	Interrupted system call
5	Input/output error
6	No such device or address
7	Argument list too long
8	Exec format error
9	Bad file descriptor
10	No child processes
11	Resource temporarily unavailable
12	Cannot allocate memory
13	Permission denied
14	Bad address
15	Block device required
16	Device or resource busy
17	File exists
18	Invalid cross-device link
19	No such device
20	Not a directory
21	Is a directory
22	Invalid argument
23	Too many open files in system
24	Too many open files
25	Inappropriate ioctl for device
26	Text file busy
27	File too large

Table 6: Linux *errno* error numbers and their meanings

Error number	Reason for error
28	No space left on device
29	Illegal seek
30	Read-only file system
31	Too many links
32	Broken pipe
33	Numerical argument out of domain
34	Numerical result out of range
35	Resource deadlock avoided
36	File name too long
37	No locks available
38	Function not implemented
39	Directory not empty
40	Too many levels of symbolic links
41	Unknown error 41
42	No message of desired type
43	Identifier removed
44	Channel number out of range
45	Level 2 not synchronized
46	Level 3 halted
47	Level 3 reset
48	Link number out of range
49	Protocol driver not attached
50	No CSI structure available
51	Level 2 halted
52	Invalid exchange
53	Invalid request descriptor
54	Exchange full
55	No anode

Table 6: Linux *errno* error numbers and their meanings

Error number	Reason for error
56	Invalid request code
57	Invalid slot
58	Unknown error 58
59	Bad font file format
60	Device not a stream
61	No data available
62	Timer expired
63	Out of streams resources
64	Machine is not on the network
65	Package not installed
66	Object is remote
67	Link has been severed
68	Advertise error
69	Srmount error
70	Communication error on send
71	Protocol error
72	Multihop attempted
73	RFS specific error
74	Bad message
75	Value too large for defined data type
76	Name not unique on network
77	File descriptor in bad state
78	Remote address changed
79	Can not access a needed shared library
80	Accessing a corrupted shared library
81	.lib section in a.out corrupted
82	Attempting to link in too many shared libraries
83	Cannot exec a shared library directly

Table 6: Linux *errno* error numbers and their meanings

Error number	Reason for error
84	Invalid or incomplete multibyte or wide character
85	Interrupted system call should be restarted
86	Streams pipe error
87	Too many users
88	Socket operation on non-socket
89	Destination address required
90	Message too long
91	Protocol wrong type for socket
92	Protocol not available
93	Protocol not supported
94	Socket type not supported
95	Operation not supported
96	Protocol family not supported
97	Address family not supported by protocol
98	Address already in use
99	Cannot assign requested address
100	Network is down
101	Network is unreachable
102	Network dropped connection on reset
103	Software caused connection abort
104	Connection reset by peer
105	No buffer space available
106	Transport endpoint is already connected
107	Transport endpoint is not connected
108	Cannot send after transport endpoint shutdown
109	Too many references: cannot splice
110	Connection timed out
111	Connection refused

Table 6: Linux *errno* error numbers and their meanings

Error number	Reason for error
112	Host is down
113	No route to host
114	Operation already in progress
115	Operation now in progress
116	Stale NFS file handle
117	Structure needs cleaning
118	Not a XENIX named type file
119	No XENIX semaphores available
120	Is a named type file
121	Remote I/O error
122	Disk quota exceeded
123	No medium found
124	Wrong medium type

Table 6: Linux *errno* error numbers and their meanings

11 Appendix—States

11.1 Basic states

Table 7 lists the states that detectors may report to the base monitor:

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

Table 7: States reported by detectors for RMS objects

Table 8 lists additional resource states that may be displayed in the Cluster Admin GUI or by `hvdisp`:

State	Description
Deact	Applies to <code>userApplication</code> and <code>objects</code> only. Operator intervention has deactivated the application throughout the cluster (such as for maintenance purposes).
Inconsistent	Applies to <code>userApplication</code> and <code>objects</code> only. The object is <code>Offline</code> or <code>Faulted</code> , but one or more resource objects in its graph have their <code>ClusterExclusive</code> attribute set to 1 and are <code>Online</code> or <code>Faulted</code> .
OfflineFault	Fault that occurred in the past has not yet been cleared.
Unknown	No information is available. Reported before object initialization is completed.

Table 8: Additional states that may be displayed for RMS objects

State	Description
Wait	Temporarily in transition to a known state. An action has been initiated for the affected resource, and the system is waiting for the action to be completed before allocating one of the above states.
Warning	Some warning threshold has been exceeded. Note that this state is reported only for selected resources.
Maintenance	<p>Manual, temporary mode of operation in which the state of an application is decoupled from the states of its dependent resources. This allows, for example, a file system to be taken offline for backup without disturbing the state of its parent application.</p> <p>An application in maintenance mode is usually marked with its intended state, which is the state that would be attained if the application were immediately taken out of maintenance mode. The maintenance mode intended states are Maintenance-Online, Maintenance-Offline, and Maintenance-Standby.</p>

Table 8: Additional states that may be displayed for RMS objects

The interpretation of `Offline` and `Faulted` may depend on the resource type. For instance, a mount point resource can be either `Online` (mounted) or `Offline` (not mounted); in this case, the detector would never report the `Faulted` state. On the other hand, a detector for a physical disk can report either `Online` (normal operation) or `Faulted` (input or output error); it would never report `Offline`.

11.2 State details

Besides the basic states listed above, RMS may report additional state details in the following locations:

- In the Cluster Admin GUI, the properties view of an object includes the *State Details* item at the top of the list. Unlike most other attributes, which are determined at configuration time by or the Wizard Tools, this information-only field is dynamically set by RMS at runtime.
- In the output of the `hvd disp` utility, the `StateDetails` column appears at the end of each line.

In most cases, the *StateDetails* field is empty. RMS typically provides this extra information when an application is in maintenance mode, or when an object is in a transitional, inconsistent, or standby state. Table 9 lists all possible *StateDetails* values for RMS objects.

Value	Description
Failed Over	Offline processing successful and failover initiated
Faulted	Received Faulted report
Inconsistent on remote	userApplication is Online on multiple hosts, but is not Online on the local host
Initial Fault	userApplication already faulted when RMS started
Joined	SysNode is in Offline state because it has joined the cluster
Killed	SysNode is in Faulted state because of a successful kill
Not Joined	SysNode is in Offline state because it has not yet joined the cluster
Offline	Received Offline report
Offline Failed	Offline processing failed
Offline Success	Offline processing successful
Offline intended	Intended state is Offline
Online	Received Online report
Online	userApplication is Online on multiple hosts

Table 9: StateDetails values for RMS objects

Value	Description
Online !!	Intended state is <i>Online</i> , but some resources have conflicting states
Online intended	Intended state is <i>Online</i>
PreCheckScriptFailed	PreCheckScript failed
Preserved	PreserveState set , no <i>Offline</i> processing initiated
Shutdown	SysNode is in Faulted state because it has been shutdown
Standby	Received <i>Standby</i> report
Standby !!	Intended state is <i>Standby</i> , but some resources have conflicting states
Standby intended	Intended state is <i>Standby</i>
Remote Faulted	Controlled application is Faulted on at least one host
Remote Offline	Controlled application is not consistently <i>Standby</i> on all hosts

Table 9: StateDetails values for RMS objects

For example, if an application was online on a particular node before it was put into maintenance mode, it will generally return to the online state on the same node when it leaves maintenance mode. RMS indicates this by reporting *Online intended* in the state details field on that node. On other nodes where the application was previously offline, RMS will report *Offline intended* in the state details field.

12 Appendix—Object types

The following alphabetical list describes all object types that are supplied with RMS and configured by or the Wizard Tools.

andOp

Required attributes:

HostName (for direct children of a userApplication object)

Object associated with its children by a logical *AND* operator. This object type is online if all children are online, and offline if all children are offline.

controller

Required attributes:

Resource

Object that allows a parent userApplication to control one or more child userApplication objects.

For backward compatibility, PCS supports legacy controller objects if they already exist in the configuration. However, they are mutually exclusive with the current gController objects.

ENV

Required attributes:

(none required)

Object containing clusterwide (global) environment variables.

ENVL

Required attributes:

(none required)

Object containing node-specific (local) environment variables.

gResource

Required attributes:

rKind

rName

Custom (generic) object. Usually represents system resources such as file systems, network interfaces, or system processes.

orOp

Required attributes:

(none required)

Object associated with its children by a logical *OR* operator. This object type is online if at least one child is online.

SysNode

Required attributes:
(none required)

Represents nodes in the cluster; at least one required. Only `userApplication` objects are allowed as its children.

userApplication

Required attributes:
(none required)

Represents an application to be monitored; at least one required. Must have one or more `SysNode` objects as its parents. For each `SysNode` parent, it must have one child `andOp` with its `HostName` attribute set to the name of the corresponding `SysNode`.

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 the Wizard Tools. The following sections list all attributes along with their possible settings and default values.

13.1 Attributes available to the user

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

The following default value is for Cmdline resource. Other resources have a default value individually.

AlternateIp

Possible Values: Any interconnect name

Default: "" (empty)

Valid for SysNode objects. Space-separated list that RMS uses as additional cluster interconnects if the interconnect assigned to the SysNode name becomes unavailable. All these interconnects must be found in the /etc/hosts database. By default, the configuration wizards assume the alternate interconnects to node <nodename> have names of the form <nodename>rmsAI<nn>, where <nn> is a two-digit, zero-filled number. This setting is restricted to very specific configurations and must never be used in a cluster with CF as interconnect.

AutoRecover

Possible Values: 0, 1

Default: 1

Valid for resource objects. If set to 1, executes the online script for an object if the object becomes faulted while in an Online state. If the object is able to return to the Online state, the fault is recovered.

This attribute must be 0 for controller objects: RMS handles switchover of child applications automatically.

AutoRecoverCleanup*Possible Values:* 0, 1*Default:* 0

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`.

AutoStartUp*Possible Values:* 0, 1*Default:* 0

Valid for `userApplication` objects. If set to 1, automatically brings the application `Online` on its highest priority `SysNode` (the first node in its `PriorityList` attribute) when RMS is started. Note that the application will not start automatically if the highest priority `SysNode` is unavailable, regardless of the state of the other nodes.

You can override the `AutoStartUp` attribute for all `userApplication` objects by setting the `HV_AUTOSTARTUP` variable. See the description of `HV_AUTOSTARTUP` in the section “Local environment variables” on page 309.

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

Valid for `userApplication` objects. Configures an application for automatic switchover if it becomes faulted. The values can be combined using the vertical bar (“|”) character. The `No` value inhibits automatic switchover and cannot be combined with any other value.

For backward compatibility, the numeric values 0 and 1 are accepted: 0 is equivalent to `No`, and 1 is equivalent to `HostFailure | ResourceFailure | ShutDown`.

ClusterExclusive*Possible Values:* 0, 1*Default:* 1

Valid for resource objects. If set to 1, guarantees that the resource is online on only one node in the cluster at any time. If set to 0, allows a resource to be online on more than one node at a time. Note that “online” in this context refers to any phase of online processing. For instance, if a

resource is in the `Online` state on one node while its `PreOnlineScript` is executing on another node, then both resource objects would be considered as online for the purposes of this test.

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

ControlledSwitch

Possible Values: 0, 1

Default: 1 in follow mode or local mode, 0 in roaming mode

Valid for controlled `userApplication` objects. If set to 0, RMS allows a manual switch request from the CLI or the GUI. If set to 1, only the parent controller can issue switch requests to this `userApplication`.

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 `Failed` state.

Halt (Wizard Tools)

Possible Values: 0, 1

Default: 0

Valid for `userApplication` objects. Controls local node elimination in the event of a double fault. A double fault occurs when a second fault is generated during the initial fault processing of an application.

If `Halt` is set to 1, and another node is available to run the application, a double fault will trigger the following sequence of events:

1. First, RMS on the local node will exit immediately.
2. Next, RMS on another node will invoke the Shutdown Facility to eliminate the local node.
3. Finally, all applications that were online on the local node, and that have their `AutoSwitchOver` parameter set to include `HostFailure`, will be switched over to the available node.

Note: Even if all the conditions for the `Halt` attribute are met for an application (`AutoSwitchOver` setting, additional hosts available), other applications running on the same host may block the `Halt` operation. For instance, another application may have no other available hosts, or it may not have the appropriate `AutoSwitchOver` setting. In either case, RMS

will continue to run on the local node. To prevent this, allocate additional hosts for the other applications and adjust their priority lists to minimize node conflicts with the application that has its `HalT` attribute set.

I_List

Possible Values: Space-separated list of `SysNode` or `names`

Default: "" (empty)

Valid for all `SysNode` and `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` or `objects` must have the same number of additional interconnects.

LieOffline

Possible Values: 0, 1

Default: 0

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

MonitorOnly

Possible Values: 0, 1

Default: 0

Valid for resource objects. If set to 1, a faulted state of the object is ignored by the parent when calculating the parent's state. A parent must 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` and `objects`. Allows RMS to start the application on the node where it was last online when the entire cluster was brought down and then restarted. If set to 0 or not set (the default), the application comes online on the node with the highest priority in the attribute `PriorityList`. If set to 1, the application comes online on the

node where it was last online. In case of `AutoStartUp` or a priority switch, this last-online node has the highest priority, regardless of its position in the priority list.

RMS keeps track of where the application was last online by means of timestamps. The node which has the latest timestamp for an application is the node on which the application will go online. Different cluster nodes should be in time-synchronization with each other, but this is not always the case. Since RMS does not provide a mechanism for ensuring time-synchronization between the nodes in the cluster, this responsibility is left to the system administrator. If RMS detects a severe time-discrepancy between the nodes in the cluster, an `ERROR` message is printed to the switchlog.

The `ntp` time service should be used to establish consistent time across the nodes in the cluster. Refer to the manual page for `ntpd` or `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`, `,` and `gController` 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 whether an application can negotiate online requests.

If set to 0, then the application can negotiate its online request only when all nodes where it can possibly run are online.

If set to 1, then the application can negotiate its online request within the current set of online nodes, even if some other nodes (including the application's primary node) are offline or faulted.

Note that a `userApplication` that has its `PartialCluster` attribute set will not be affected by startup timeouts from remote nodes: the application can still go online on the local node. See the description of `HV_AUTOSTART_WAIT` in the section "Global environment variables" on page 304.

PersistentFault*Possible Values:* 0, 1*Default:* 0

Valid for `userApplication` and `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` or removed from the configuration.

PostOfflineScript*Possible Values:* Valid script (character)*Default:* "" (empty)

Valid for all objects except `SysNode` and `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` and `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` and `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` and `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` and `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 `controller` object, the order of nodes in `PriorityList` is ignored. However, each child application must be able to run on the nodes specified for the parent application.

Resource

Possible Values: Valid name (character)

Default: "" (empty)

Valid for `controller` objects. Contains the name of the child (controlled) `userApplication`.

ScriptTimeout

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

Default: 300

Valid for all object types. Specifies the timeout value for all scripts associated with that object in the configuration file. RMS sends a kill signal to the script if the timeout expires.

Use the string format to specify individual timeout values of `offline_value` for `OfflineScript` and `online_value` for `OnlineScript`.

ShutdownPriority

Possible Values: 0–`MAXINT`

Default: 0

Valid for `userApplication` and `objects`. `ShutdownPriority` assigns a weight factor to the application for use by the Shutdown Facility.

When interconnect failures and the resulting concurrent node elimination requests occur, SF calculates the shutdown priority of each subcluster as the sum of the subcluster's SF node weights plus the RMS ShutdownPriority of all online application objects in the subcluster. The optimal subcluster is defined as the fully connected subcluster with the highest weight.

StandbyCapable

Possible Values: 0, 1

Default: 0

Valid for resource objects. If set to 1, the object performs standby processing on all nodes where the parent application is supposed to be Offline.

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

StandbyTransitions

Possible Values: `Startup`, `SwitchRequest`, `ClearFaultRequest` or any combination joined by vertical bars (|)

Default: "" (empty)

Valid for `userApplication` and `objects`. The value specifies when standby processing is initiated for the application object:

- `Startup`—at startup. This setting is ignored if the real-world application is already online, or if the application object is forced to go online because the `AutoStartup` attribute is set.
- `SwitchRequest`—after application switchover, if the application was online before the switchover.
- `ClearFaultRequest`—after a faulted state is cleared with `'hvutil -c'`.

WarningScript

Possible Values: Valid script (character)

Default: "" (empty)

Valid for GDS resource objects. 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 or by RMS at runtime.

Affiliation

Possible Values: Any string

Default: "" (empty)

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

Class

Possible Values: any string

Default: Default type defined in the chapter “Appendix—Object types” on page 289.

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

Comment

Possible Values: any string

Default: "" (empty)

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

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.

Note that a `controller` object has no detector because RMS determines its state internally.

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.

LastDetectorReport

Possible Values: Online, Offline, Faulted, Standby

Default: (none)

Valid for resource objects with detector. This attribute contains the most recent detector report for the object. The value may be displayed in the Cluster Admin GUI; the possible values depend on the type of resource the object represents.

MaxControllers

Possible Values: 0–512

Default: 512

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

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 with 'hvdisp -S <resource_name>'.

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 use with dynamic reconfiguration only. NullDetector must never be set hardcoded to on in the RMS configuration file.

OfflineDoneScript

Possible Values: Valid script (character)

Default: "" (empty)

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

PreCheckScript

Possible Values: Valid script (character)

Default: "" (empty)

Valid for `userApplication` and `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.

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 `gController` and `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 `IgnoreOfflineRequest` and `IgnoreOnlineRequest` are respectively set to 0.

StateDetails

Possible Values: Any string

Default: "" (empty)

Valid for all objects. Displays additional state details in the Cluster Admin GUI or the `hvdsp` CLI user interface. In most cases, the state details field is empty. RMS typically provides this extra information when an application is in maintenance mode, or when an object is in a transitional, inconsistent, or standby state.

14 Appendix—Environment variables

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

- “Global environment variables” on page 304
- “Local environment variables” on page 309
- “Script execution environment variables” on page 313

14.1 Setting environment variables



Caution

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



Do not change the `hvenv` configuration file. Changes to your configuration's environment variables should be confined to the `<RELIANT_PATH>hvenv.local` file.

The values of environment variables are specified as `export` directives in the `hvenv.local` file. To adjust a variable's setting, you would open `hvenv.local` with a text editor of your choice and modify (or add) the appropriate line.

A typical `export` directive would appear as follows:

```
export SCRIPTS_TIME_OUT=200
```

When RMS starts, it reads the values of environment variables from `hvenv` and `hvenv.local` and initializes the `ENV` and `ENVL` objects respectively. No further reference is made to these two configuration files while RMS is running. Therefore, any changes you make to `hvenv.local` will not take effect until the next time RMS starts up.

Values in the `ENVL` (local) object override values in the `ENV` (global) object. If a global variable setting appears in the `hvenv.local` file, it will override the corresponding setting in the `hvenv` file. However, if you adjust a global variable in the `hvenv.local` file on one node, you must make the same adjustment to `hvenv.local` on every other node in the cluster. Global variable settings must agree clusterwide.

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

- `hvdisp ENV`
- `hvdisp ENVL`

14.2 Global environment variables



Global variable settings (`ENV`) are included in the configurations checksum that is common to the cluster. The checksum is verified on each node during startup of the base monitor. RMS will fail to start if it detects a checksum difference between the values on any two nodes.



The default values of the environment variables are found in `<RELIANT_PATH>/bin/hvenv`. They can be redefined in the `hvenv.local` configuration file.

The following list describes the global environment variables for RMS:

`HV_AUTOSTARTUP_IGNORE`

Possible values: List of RMS cluster nodes. The list of RMS cluster nodes must be the names of the `SysNodes` as found in the RMS configuration file. The list of nodes cannot include the CF name.

Default: "" (empty)

List of cluster nodes that RMS ignores when it starts. This environment variable is not set by default. A user application will begin its automatic startup processing if the `AutoStartUp` attribute is set and when all cluster nodes defined in the user application have reported `Online`. If a cluster node appears in this list, automatic startup processing will begin even if this node has not yet reported the `Online` state.

Use this environment variable if one or more cluster nodes need to be taken out of the cluster for an extended period and RMS will continue to use the configuration file that specifies the removed cluster nodes. In this case, specifying the unavailable cluster nodes in this environment variable ensures that all user applications are automatically brought online even if the unavailable cluster nodes do not report `Online`.



Caution

If the `HV_AUTOSTARTUP_IGNORE` environment variable is used, ensure that it is correctly defined on all cluster nodes and that it is always kept up-to-date. When a node is brought back into the

cluster, remove it from this environment variable. If this does not occur, data loss could occur because RMS will ignore this node during the startup procedure and will not check whether the application is already running on the nodes specified in this list. It is the system administrator's responsibility to keep this list up-to-date if it is used.

HV_AUTOSTART_WAIT

Possible values: 0–MAXINT

Default: 60 (seconds)

Defines the period (in seconds) that RMS waits for cluster nodes to report `Online` when RMS is started. If this period expires and not all cluster nodes are online, a switchlog message indicates the cluster nodes that have not reported `Online` and why the user application(s) cannot be started automatically.

Note that `HV_AUTOSTART_WAIT` timeouts from remote nodes will not affect a local `userApplication` that has its `PartialCluster` attribute set: the application can still go online on the local node. See the description of the `PartialCluster` attribute in the section “Attributes available to the user” on page 291.



This attribute generates a warning message only. `AutoStartUp` will proceed even if the specified period has expired.

HV_CHECKSUM_INTERVAL

Possible values: 0–MAXINT

Default: 120 (seconds)

Interval in seconds for which the RMS base monitor waits for each `Online` node to verify that its checksum is the same as the local checksum.

If checksums are confirmed within this interval, then RMS on the local node continues its operations as usual. However, if a checksum from a remote node is not confirmed, or if it is confirmed to be different, then the local monitor shuts down if it has been started less than `HV_CHECKSUM_INTERVAL` seconds before.

Also, if a checksum from a remote node is not confirmed, or if the checksum is confirmed to be different, then the local monitor considers the remote node as `Offline` if that local monitor has been started more than `HV_CHECKSUM_INTERVAL` seconds before.

HV_COM_PORT

Possible values: 0–MAXINT

Default: 8000

The communication port used by the RMS base monitor on all nodes in the cluster.

HV_LOG_ACTION_THRESHOLD

Possible values: 0–100

Default: 98

Determines when `hvllogcontrol` takes action to clean up RMS log files. If the percentage of used space on the file system containing `RELIANT_LOG_PATH` is greater than or equal to this threshold, all subdirectories below `RELIANT_LOG_PATH` will be removed. Furthermore, if `HV_LOG_ACTION` is set to `on` and all subdirectories have already been removed, the current log files will be removed too. See “`HV_LOG_ACTION`” on page 309 for more information.

HV_LOG_WARN_THRESHOLD

Possible values: 0–100

Default: 95

Defines when `hvllogcontrol` warns the user about the volume of RMS log files. If the percentage of used space on the file system containing `RELIANT_LOG_PATH` is greater than or equal to this threshold value, `hvllogcontrol` issues a warning to the user. See also `HV_LOG_ACTION_THRESHOLD` above.

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_USE_ELM

Possible values: 0, 1

Default: 1

Specifies the heartbeat monitoring mode used by the RMS base monitor:

0—remote node and base monitor states are detected by periodically sending UDP heartbeat packets across the network. If no heartbeats are received from a remote node during an interval defined by `HV_CONNECT_TIMEOUT`, RMS marks the node as down and waits for a recovery period before taking further action.

1—combines the Enhanced Lock Manager (ELM) method and the UDP heartbeat method. This setting is valid only when CF is installed and configured. The ELM lock is taken and held by the local node until ELM reports a remote node down or remote base monitor down. In either of these cases, the remote node is immediately killed. Until ELM reports a change in a remote node's state, RMS also monitors the UDP heartbeat of each remote node as described above, but with a much longer recovery timeout.

Whether or not ELM is enabled, a remote node is killed if its UDP heartbeat is not received before its heartbeat recovery timeout expires. When CF is not present, ELM is disabled automatically, and the heartbeat recovery timeout defaults to 45 seconds. When CF is present, ELM is enabled by default, and the heartbeat recovery timeout defaults to 600 seconds; this avoids premature node kills when the remote node is slow to respond.

Only experts should disable ELM manually. When CF is present but ELM is disabled, the default 600 second heartbeat recovery timeout is too long for efficient detection of remote RMS or node outages. In this case, the recovery timeout on the local node must also be adjusted manually by starting RMS with the `'hvcn -h <timeout> -c <config_file>'` command. Note that the recovery timeout should be set to the same value on every node in the cluster. When ELM is disabled, the recommended global value is 45 seconds.

RELIANT_LOG_LIFE

Possible values: Any number of days

Default: 7 (days)

Specifies the number of days that RMS logging information is retained. Every time RMS starts, the system creates a directory that is named on the basis of when RMS was last started, and which contains all

constituent log files. All RMS log files are preserved in this manner. All log files which are older than the number of days specified in this variable are deleted by a `cron` job.

RELIANT_LOG_PATH

Possible values: Any valid path

Default: `/var/opt/SMAWRrms/Log`

Specifies the directory where all RMS, PCS, and Wizard Tools log files are stored.

RELIANT_PATH

Possible values: Any valid path

Default: `/opt/SMAW/SMAWRrms`

Specifies the root directory of the RMS directory hierarchy. Users do not normally need to change the default setting.

RELIANT_SHUT_MIN_WAIT

Possible values: `0-MAXINT`

Default: 900 (seconds)

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

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

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

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



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

14.3 Local environment variables

Local environment variable settings can vary from node to node. The following list describes the local environment variables for RMS:

HV_AUTOSTARTUP

Possible values: 0, 1

Default: 1 (normal processing of `AutoStartUp` attribute)

Controls the action of the `AutoStartUp` attribute for all `userApplication` objects on the local node. If set to 1 (the default value) the automatic startup of each `userApplication` is determined by its `AutoStartUp` attribute (see the section “Attributes available to the user” on page 291). If set to 0, the `AutoStartUp` attribute is ignored and no automatic startup occurs. `HV_AUTOSTARTUP` can be set in the Cluster Admin *Tools* menu or by using the `hvsetenv` command; in either case, the change does not take effect until the next RMS startup.

HV_CONNECT_TIMEOUT

Possible values: 5–`MAXINT`

Default: RHEL-AS: 5 (seconds), RHEL5 : 30 (seconds). Users do not normally need to change the default setting.

The maximum time (in seconds) that the heartbeat from a node is not received before the base monitor assumes the connection to that node has been lost and starts the UDP heartbeat recovery timer.

Input values less than 5 are converted internally to 5.

HV_LOG_ACTION

Possible values: on, off

Default: off

Determines whether the current log files in the *RELIANT_LOG_PATH* directory will be deleted when the percentage of used space on the file system containing *RELIANT_LOG_PATH* is greater than or equal to *HV_LOG_ACTION_THRESHOLD*. See “*HV_LOG_ACTION_THRESHOLD*” on page 306 for more information.

HV_MAX_HVDISP_FILE_SIZE

Possible values: 0–MAXINT

Default: 20,000,000 (bytes)

Prevents the unlimited growth of the temporary file that RMS uses to supply *hvdisp* with configuration data and subsequent configuration and state changes. The value of this variable is the maximum size in bytes of the temporary file *<RELIANT_PATH>/locks/.rms.<process id of the hvdisp process>*.

HV_MAXPROC

Possible values: 0–fork limit

Default: 30

Defines the maximum number of scripts RMS can have forked at any time. The default (30) is sufficient in most cases.

HV_MLOCKALL

Possible values: 0, 1

Default: 0

If set to 1, the base monitor process and any memory it allocates will be locked in memory. If set to 0 (the default), the base monitor may be swapped out.

HV_RCSTART

Possible values: 0, 1

Default: 1 (start RMS in the *rc* script)

Determines if RMS is started in the *rc* script. If set to 1 (the default value), RMS is started automatically at system boot time. If set to 0, RMS must be started manually. *HV_RCSTART* can be set in the Cluster Admin *Tools* menu or by using the *hvsetenv* command. (Prerequisite for *rc* start: *CONFIG.rms* exists and contains a valid entry.)

HV_REALTIME_PRIORITY

Possible values: 1–99

Default: 50

Defines the real time priority for the RMS base monitor and its detectors. Caution should be used when adjusting this variable. High settings can prevent other OS real-time processes from getting their processor time slice. Low settings can prevent the RMS base monitor from reacting to detector reports and from performing requests from command line utilities.

This variable is processed only on Solaris platforms. It has no effect on Linux platforms.

HV_SCRIPTS_DEBUG

Possible values: 0, 1

Default: 0

Controls debugging output from RMS scripts. If this variable is set to 1, it overrides the setting for scripts that are generated and managed by or the Wizard Tools, causing them to write detailed runtime information about the commands that are executed to the RMS `switchlog` file. The type of information logged may vary according to the script. This setting applies only to those scripts provided with PRIMECLUSTER products. To disable script debug message logging, delete the `HV_SCRIPTS_DEBUG` entry or set `HV_SCRIPTS_DEBUG=0` in `hvenv.local`.

Note: when this variable appears in `hvenv.local`, RMS adds it to the script environment but otherwise makes no attempt to process it. Therefore, it is not reported in the Cluster Admin GUI or in 'hvdisp ENVL' output.

HV_SYSLOG_USE

Possible values: 0, 1

Default: 1 (in `hvenv`)

Controls output to the system log from the RMS base monitor. RMS always records RMS ERROR, FATAL ERROR, WARNING, and NOTICE messages in the RMS `switchlog` file. By default, these messages are duplicated in the system log file `/var/adm/messages` (Solaris) or `/var/log/messages` (Linux). To disable RMS messages in the system log, set `HV_SYSLOG_USE=0` in `hvenv.local`.

RELIANT_HOSTNAME

Possible values: valid name

Default: <nodename>RMS

The name of the local node in the RMS cluster. The default value of this variable is the node name with an RMS suffix (for example: `shasta1RMS`), as generated by the following command:

```
export RELIANT_HOSTNAME=`cftool -l 2>/dev/null | \
tail -1 | cut -f1 -d" "`RMS
```

If CF is not installed, this variable is generated by the following command:

If this preset value is not suitable, it must be modified accordingly on all nodes in the cluster.

The specified cluster node name must correspond to the `SysNode` name in the `<configname>.us` configuration file. The node name determines the IP address that RMS uses for establishing contact with this node.

RELIANT_INITSCRIPT

Possible values: any executable

Default: `<RELIANT_PATH>/bin/InitScript`

Specifies an initialization script to be run by RMS when the system is started. This script is run before any other processes are activated. It is a global script that is run once on every cluster node on which it is defined.

RELIANT_STARTUP_PATH

Possible values: any valid path

Default: `<RELIANT_PATH>/build`

Defines where RMS searches at start time for the configuration files.

SCRIPTS_TIME_OUT

Possible values: 0–MAXINT

Default: 300 (seconds)

Specifies the global period (in seconds) within which all RMS scripts must be terminated. If a specific script cannot be terminated within the defined period, it is assumed to have failed and RMS begins appropriate processing for a script failure.

If this value is too low, error conditions will be produced unnecessarily, and it may not be possible for the applications to go online or offline. An excessively high value is unsuitable because RMS will wait for this period to expire before assuming that the script has failed.

In case the global setting is not appropriate for all objects monitored by RMS, this global value can be overridden by an object-specific setting of the `ScriptTimeout` attribute.

14.4 Script execution environment variables

The variables in this section are set by the RMS base monitor when it executes an object's script. These exist only in the script's environment and only for the duration of the script execution. Since these variables are explicitly set, they have no default values.

HV_APPLICATION

Possible values: any userApplication name

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

HV_AUTORECOVER

Possible values: 0, 1

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

HV_FORCED_REQUEST

Possible values: 0, 1

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

HV_LAST_DET_REPORT

Possible values: one of Online, Offline, Faulted, NoReport

Last detector report for the current object.

HV_OFFLINE_REASON

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

Reason for ongoing offline processing:

DEACT: deact request ('hvutil -d')

SWITCH: manual switchover ('hvswitch')

FAULT: follow-up processing after a previous resource failure

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

HV_NODENAME

Possible values: any object name

Name of current object.

HV_REQUESTING_CONTROLLER

Possible values: controller name and node name

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

HV_SCALABLE_CONTROLLER

Possible values: scalable controller name

The name of the scalable controller that controls this application.

HV_SCALABLE_INFO

Possible values: scalable applications list

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

HV_SCRIPT_TYPE

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

Script type.

NODE_SCRIPTS_TIME_OUT

Possible values: 0–MAXINT

Timeout value for the current object and script type.



HV_REQUESTING_CONTROLLER, HV_SCALABLE_CONTROLLER and HV_SCALABLE_INFO can be used only by the Online/Offline processing of the application controlled with the controller object.

15 Appendix—RMS command line interface

The primary interface for configuring RMS is or the RMS Wizard Tools, and the primary interface for administering RMS is the Cluster Admin GUI. These user interfaces call the RMS command line interface (CLI), and, under certain conditions, you may find it useful to invoke the CLI directly.

The following section lists the RMS CLI commands available to administrators. For a complete description of any command's usage, see its [online man page](#). For a list of all PRIMECLUSTER commands related to RMS, see the chapter “Appendix—List of manual pages” on page 319.



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



RMS CLI commands accept case-insensitive application names.

15.1 Available RMS CLI commands

`hvassert`

Tests an RMS resource for a specified resource state. It can be used in scripts when a resource must achieve a specified state before the script can issue the next command. Does not require root privilege.

`hvattr`

Provides an 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 `HostFailure`, `ResourceFailure`, `ShutDown`, or `No` may be specified. For more information, see the description of the `AutoSwitchOver` attribute in “Appendix—Attributes” on page 291.

`hvcm`

Starts the base monitor and the detectors for all monitored resources. In most cases, it is not necessary to specify options to the `hvcm` command.

The base monitor is the decision-making module of RMS. It controls the configuration and access to all RMS resources. If a resource fails, the base monitor analyzes the failure and initiates the appropriate action according to the specifications for the resource in the configuration file.

hvconfig

Either displays the current RMS configuration or sends the current configuration to an output file.

The output of the `hvconfig` command is equivalent to the running RMS configuration file, but does not include any comments that are in the original file. Also, the order in which the resources are listed in the output might vary from the actual configuration file.

hvdisp

Displays information about the current configuration for RMS resources. Does not require root privilege.

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.

hvlogclean

Either saves old log files into a subdirectory whose name is the time RMS was last started, or, if invoked with the `-d` option, deletes old log files. In either case, `hvlogclean` creates a clean set of log files even while RMS is running.

hvrclv

On Solaris platforms, displays or changes the run level used for RMS when it is started automatically at system startup. During installation, `pkgadd` uses `hvrclv` to set the RMS run level to the default in `/etc/inittab`. If the system default run level is changed at a later time, the RMS run level should be adjusted accordingly with `hvrclv` to ensure that RMS starts in the proper sequence. `hvrclv` may also be used to display the current system default run level.

hvreset

Reinitializes the graph of an RMS user application on one or more nodes in the configuration. Running scripts will be terminated, ongoing requests and contracts will be cleaned up, and information about previous failures will be purged. If the process is successful, the entire graph will be brought back into a consistent initial state, but an inconsistent state is also a possible result. Therefore, use this command for test purposes only, and never invoke it on a production cluster.

**Caution**

This command is intended for use by experts only.

hvsetenv

Provides an interface for changing the following RMS environment variables on the local node:

- `HV_RCSTART` controls the automatic startup of RMS.
- `HV_AUTOSTARTUP` controls the automatic startup of all applications.

For more information about these environment variables, see “Appendix—Environment variables” on page 303.

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 by creating queues for sequential processing.

hvutil

Provides general administration interface to RMS. It performs various resource administration tasks, such as dynamically setting logging levels, sending a resource `Offline`, clearing faulted resources or hung cluster nodes in the `Wait` state, setting detector time periods, setting Maintenance Mode, and so forth.

**Caution**

Setting high logging levels in `hvutil` can cause disk overflow if enabled for too long. See the online man pages for more information.

16 Appendix—List of manual pages

This appendix lists the online manual pages for CCBR, CF, CIP, PAS, Resource Database, RMS, SF, Monitoring Agent, SIS, Web-Based Admin View, RMS Wizards, and miscellaneous utilities. To display a manual page, enter the following command:

```
$ man man_page_name
```



This is not an exhaustive list. Some PRIMECLUSTER utilities are intended only for internal use, so their manual pages are not listed here.

16.1 CCBR

System administration

cfbackup

save the cluster configuration information for a PRIMECLUSTER node

cfrestore

restore saved cluster configuration formation on a PRIMECLUSTER node

16.2 CF

System administration

cfconfig

configure or unconfigure a node for a PRIMECLUSTER cluster

cfset

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

cfrecon

dynamically reconfigure the cluster interconnects used by a node

cftool

print node communications status for a node or the cluster

`rcqconfig`

configures or reports cluster quorum settings

`rcquery`

acquires the state of consistency (quorum) of the cluster

16.3 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

16.4 PAS

System administration

`mipcstat`

MIPC statistics

`clmstat`

CLM statistics

16.5 Resource Database



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

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
- clinitscript**
report the connection confirmation results for shared disk units
- clrestorerdb**
restore the resource database
- clsetacfparam**
checks the connections of shared disk units and sets up the operation for automatic resource registration
- clsetparam**
display and change the resource database operational environment
- clsetup**
set up the resource database
- clspconfig**
set up the operation of patrol diagnosis
- clsp1**
execute patrol diagnosis
- clstarttrsc**
resource activation
- clstoprsc**
resource deactivation
- clsyncfile**
distribute a file between cluster nodes

User command

- clgettree**
display the tree information of the resource database

16.6 RMS

System administration

- hvassert**
assert (test for) an RMS resource state
- hvattrib**
make clusterwide attribute changes at runtime from a single node (installed with or 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 (for use by experts in test conditions only—not for use on production clusters)
- hvsetenv**
controls automatic startup of RMS or all user applications on the local host
- hvshut**
shut down RMS
- hvswitch**
switch control of an RMS user application resource to another node

hvtrottle

prevent multiple RMS scripts from running simultaneously

hvutil

manipulate availability of an RMS resource

File formats**config.us**

format of RMS node configuration file

hvenv.local

RMS local environment configuration file

16.7 SF

System administration**rcsd**

shutdown daemon for the Shutdown Facility

sdtool

interface tool for the shutdown daemon

File formats**rcsd.cfg**

configuration file for the shutdown daemon

SA_blade.cfg

configuration file for FTS server blade shutdown agent

SA_ipmi.cfg

configuration file for the BMC (Board Management Controller) shutdown agent on IPMI (Intelligent Platform Management Interface) compliant platforms

SA_pprci.cfg

configuration file for RCI shutdown agent (PRIMEPOWER only)

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_SATxscf.cfg`
configuration file for Solaris midrange server XSCF board shutdown agent
- `SA_SATxscf2.cfg`
configuration file for Solaris APL server XSCF board shutdown agent
- `SA_scon.cfg`
configuration file for SCON shutdown agent
- `SA_snmp.cfg`
configuration file for SNMP shutdown agent
- `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



To see the system administration man pages for any of the above shutdown agents, omit the '.cfg' suffix.

16.8 Monitoring Agent

System administration

- `cldevparam`
Changes and displays the tunable operation environment for asynchronous monitoring
- `clmbmonctl`
Starts, stops, restarts, and displays the operating system of the MMB asynchronous monitoring daemon
- `clmbsetup`
Registers, changes, deletes, and displays MMB information

`clrcimonctl`

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

`clrcumonctl`

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

`clrcusetup`

Registers, changes, deletes, or displays console information

16.9 SIS

System administration

`dtcpadmin`

start the SIS administration utility

`dtcpd`

start the SIS daemon for configuring VIPs

`dtcpstat`

status information about SIS

16.10 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

16.11 RMS Wizards

RMS Wizard Tools and RMS Wizard Kit

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

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

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

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

System administration

`clrwconfig`

set up the linking function between the PRIMECLUSTER resource manager and the middleware products after the RMS configuration definitions are activated

16.12 Miscellaneous utilities

`cluster_uninstall`

uninstall PRIMECLUSTER software (man page available only for Solaris)

`sshconf`

configure `ssh` access between two or more nodes

Glossary

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

- (CF)—Cluster Foundation
- (PCS)—PRIMECLUSTER Configuration Services
- (RMS)—Reliant Monitor Services
- (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 Wizard Tools, or the CLI.

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

administrative LAN

An optional private local area network (LAN) used for administrative commands to the nodes in the cluster. To provide an extra level of security, normal users do not have access to the administrative LAN. In PRIMECLUSTER configurations, the System Console and Cluster Console reside on the administrative LAN if one is present.

See also public LAN.

API

See Application Program Interface.

application (RMS)

In the RMS context, an application object is a special resource used to group other resources into a logical collection. Typically, it is used to represent a real-world application or application suite in a high-availability configuration.

Application Program Interface

A shared boundary between a service provider and the application that uses that service.

application template (RMS)

A predefined group of object definition value choices used by PCS, the Wizard Tools, or the PCS Wizard Kit to create object definitions for a specific type of application.

attribute (RMS)

The part of an object definition that specifies how the base monitor acts and reacts for a particular object type during normal operations.

automatic switchover (RMS)

The procedure by which RMS automatically switches control of a `userApplication` over to another node after specified conditions are detected.

See also directed switchover (RMS), failover (RMS, SIS), switchover (RMS), symmetrical switchover (RMS).

availability

Availability describes the need of most enterprises to operate applications via the Internet 24 hours a day, 7 days a week. The relationship of the actual to the planned usage time determines the availability of a system.

base cluster foundation (CF)

This PRIMECLUSTER module resides on top of the basic OS and provides internal interfaces for the CF (Cluster Foundation) functions that the PRIMECLUSTER services use in the layer above.

See also Cluster Foundation (CF).

base monitor (RMS)

The RMS module that maintains the availability of resources. The base monitor is supported by daemons and detectors. Each node being monitored has its own copy of the base monitor.

Cache Fusion

The improved interprocess communication interface in Oracle 9i that allows logical disk blocks (buffers) to be cached in the local memory of each node. Thus, instead of having to flush a block to disk when an update is required, the block can be copied to another node by passing a message on the interconnect, thereby removing the physical I/O overhead.

CCBR

See Cluster Configuration Backup and Restore.

CF

See Cluster Foundation (CF).

CF node name (CF)

The CF cluster node name, which is configured when a CF cluster is created.

child (RMS)

A resource defined in the configuration file that has at least one parent. A child can have multiple parents, and can either have children itself (making it also a parent) or no children (making it a leaf object).

See also resource (RMS), object (RMS), parent (RMS).

cluster

A set of computers that work together as a single computing source. Specifically, a cluster performs a distributed form of parallel computing.

See also RMS configuration (RMS).

Cluster Admin

A Java-based, OS-independent management tool for PRIMECLUSTER products such as CF, SIS, RMS and PCS. Cluster Admin is available from the Web-Based Admin View interface.

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

Cluster Configuration Backup and Restore

CCBR provides a simple method to save the current PRIMECLUSTER configuration information of a cluster node. It also provides a method to restore the configuration information.

Cluster Foundation (CF)

The set of PRIMECLUSTER modules that provides basic clustering communication services.

See also base cluster foundation (CF).

cluster interconnect (CF)

The set of private network connections used exclusively for PRIMECLUSTER communications.

Cluster Join Services (CF)

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

Configuration Definition Language (PCS)

The syntax for PCS configuration templates.

See also PRIMECLUSTER Configuration Services (PCS).

configuration file (RMS)

In the RMS context, the single file that defines the monitored resources and establishes the interdependencies between them. The default name of this file is `config.us`.

console

See single console.

custom detector (RMS)

See detector (RMS).

custom type (RMS)

See generic type (RMS).

daemon

A continuous process that performs a specific function repeatedly.

database node (SIS)

Nodes that maintain the configuration, dynamic data, and statistics in a SIS configuration.

See also gateway node (SIS), service node (SIS), Scalable Internet Services (SIS).

detector (RMS)

A process that monitors the state of a specific object type and reports a change in the resource state to the RMS base monitor.

DHCP

Dynamic Host Control Protocol. A standard method of delivering information to a host at boot time. This is most often used to dynamically assign the host's IP address and netmask, but many other parameters are possible, including domain names, DNS servers, and time servers.

directed switchover (RMS)

The RMS procedure by which an administrator switches control of a `userApplication` over to another node.

See also automatic switchover (RMS), failover (RMS, SIS), switchover (RMS), symmetrical switchover (RMS).

distributing a configuration (RMS)

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

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

DOWN (CF)

A node state that indicates that the node is unavailable (marked as down). A `LEFTCLUSTER` node must be marked as `DOWN` before it can rejoin a cluster.

See also `UP` (CF), `LEFTCLUSTER` (CF), node state (CF).

Enhanced Lock Manager (ELM) (CF)

A light weight, high performance, highly responsive lock manger, specifically designed for providing a high reliability heartbeat messaging mechanism for `PRIMECLUSTER` modules.

ENS (CF)

See Event Notification Services (CF).

environment variables

Variables or parameters that are defined globally.

error detection (RMS)

The process of detecting an error. For RMS, this includes initiating a log entry, sending a message to a log file, or making an appropriate recovery response.

Event Notification Services (CF)

This PRIMECLUSTER module provides an atomic-broadcast facility for events.

failover (RMS, SIS)

With SIS, this process switches a failed node to a backup node. With RMS, this process is known as switchover.

See also automatic switchover (RMS), directed switchover (RMS), switchover (RMS), symmetrical switchover (RMS).

gateway node (SIS)

Gateway nodes have an external network interface. All incoming packets are received by this node and forwarded to the selected service node, depending on the scheduling algorithm for the service.

See also service node (SIS), database node (SIS), Scalable Internet Services (SIS).

GDS

See Global Disk Services.

generating a configuration (RMS)

The process of creating a single configuration file that can be distributed to all nodes in the configuration and activated at a later time. 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).

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).

GFS

See Global File 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.

GLS

See Global Link Services.

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).

Intelligent Platform Management Interface

A firmware and hardware specification that provides common interfaces for monitoring and managing computers. IPMI operates through an onboard Baseboard Management Controller (BMC) on the target machine to provide OS-independent remote management functions, whether or not the target machine is powered on.

interconnect (CF)

See cluster interconnect (CF).

Internet Protocol address

A numeric address that can be assigned to computers or applications.

See also *IP aliasing*.

Internode Communications facility

This module is the network transport layer for all PRIMECLUSTER internode communications. It interfaces by means of OS-dependent code to the network I/O subsystem and guarantees delivery of messages queued for transmission to the destination node in the same sequential order unless the destination node fails.

IP address

See *Internet Protocol address*.

IP aliasing

This enables several IP addresses (aliases) to be allocated to one physical network interface. With IP aliasing, the user can continue communicating with the same IP address, even though the application is now running on another node.

See also *Internet Protocol address*.

IPMI

See *Intelligent Platform Management Interface*.

JOIN (CF)

See *Cluster Join Services (CF)*.

keyword

A word that has special meaning in a programming language. For example, in an RMS configuration file, the keyword `object` identifies the kind of definition that follows.

leaf object (RMS)

A bottom object in a system graph. In the configuration file, this object definition is at the beginning of the file. A leaf object does not have children.

LEFTCLUSTER (CF)

A node state that indicates that the node cannot communicate with other nodes in the cluster. That is, the node has left the cluster. The reason for the intermediate LEFTCLUSTER state is to avoid the network partition problem.

See also *UP (CF)*, *DOWN (CF)*, *network partition (CF)*, *node state (CF)*.

link (RMS)

Designates a child or parent relationship between specific resources.

local area network

See *public LAN*.

local node

The node from which a command or process is initiated.

See also *remote node*, *node*.

log file

The file that contains a record of significant system events or messages. The ASCC control and satellite daemons maintain log files on every node on which they run. The Wizard Tools, PCS, the RMS base monitor, and RMS detectors each maintain their own log files as well.

Management Information Base

A hierarchical database of information about the local network device. The database is maintained by network management software such as an SNMP agent.

See also *Simple Network Management Protocol*.

MDS

See *Meta Data Server*.

message

A set of data transmitted from one software process to another process, device, or file.

message queue

A designated memory area which acts as a holding place for messages so they can be processed in the same order they were received.

Meta Data Server

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

MIB

See *Management Information Base*.

MMB

Abbreviation for Management Board, which is one of the hardware units installed in PRIMEQUEST.

mount point

The point in the directory tree where a file system is attached.

multihosting

Multiple controllers simultaneously accessing a set of disk drives.

native operating system

The part of an operating system that is always active and translates system calls into activities.

network partition (CF)

This condition exists when two or more nodes in a cluster cannot communicate over the interconnect; however, with applications still running, the nodes can continue to read and write to a shared device, compromising data integrity.

node

A host that is a member of a cluster.

node state (CF)

Every node in a cluster maintains a local state for every other node in that cluster. The node state of every node in the cluster must be either UP, DOWN, or LEFTCLUSTER.

See also *UP (CF)*, *DOWN (CF)*, *LEFTCLUSTER (CF)*.

object (RMS)

A representation of a physical or virtual resource in the RMS configuration file or in a system graph.

See also *leaf object (RMS)*, *object definition (RMS)*, *object type (RMS)*.

object definition (RMS)

An entry in the configuration file that identifies a resource to be monitored by RMS. Attributes included in the definition specify properties of the corresponding resource.

See also *attribute (RMS)*, *object type (RMS)*.

object type (RMS)

A category of similar resources monitored as a group, such as disk drives. Each object type has specific properties, or attributes, which limit or define what monitoring or action can occur. When a resource is associated with a particular object type, attributes associated with that object type are applied to the resource.

See also *generic type (RMS)*.

online maintenance

The capability of adding, removing, replacing, or recovering devices without shutting or powering off the node.

operating system dependent (CF)

This module provides an interface between the native operating system and the abstract, OS-independent interface that all PRIMECLUSTER modules depend upon.

Oracle Real Application Clusters (RAC)

Oracle RAC allows access to all data in a database to users and applications in a clustered or MPP (massively parallel processing) platform. Formerly known as Oracle Parallel Server (OPS).

OSD (CF)

See *operating system dependent (CF)*.

parent (RMS)

An object in the RMS configuration file or system graph that has at least one child.

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

PCS

See *PRIMECLUSTER Configuration Services (PCS)*.

PCS Wizard Kit (PCS)

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

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

primary node (RMS)

The default node on which a user application comes online when RMS is started. This is always the node name of the first child listed in the `userApplication` object definition.

PRIMECLUSTER Configuration Services (PCS)

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

PRIMECLUSTER services (CF)

Service modules that provide services and internal interfaces for clustered applications.

private network addresses

Private network addresses are a reserved range of IP addresses specified by the Internet Corporation for Assigned Names and Numbers (ICANN). Modern switches and routers prevent these addresses from being routed to the Internet, allowing two or more organizations to assign the same private addresses for internal use without causing conflicts or security risks.

private resource (RMS)

A resource accessible only by a single node and not accessible to other RMS nodes.

See also *resource (RMS)*, *shared resource*.

public LAN

The local area network (LAN) by which normal users access a machine.

See also *administrative LAN*.

queue

See *message queue*.

redundancy

The capability of one component to assume the resource load of another physically similar component in case the original component fails or is shut down. Common examples include RAID hardware and/or RAID software to replicate data stored on secondary storage devices, multiple network connections to provide alternate data paths, and multiple nodes that can be dynamically reprovisioned to maintain critical services in a cluster.

Reliant Monitor Services (RMS)

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

remote node

A node that is accessed through a LAN or telecommunications line.

See also *local node*, *node*.

reporting message (RMS)

A message that a detector uses to report the state of a particular resource to the base monitor.

resource (RMS)

A hardware or software element (private or shared) that provides a function such as a mirrored disk, mirrored disk pieces, or a database server. A local resource is monitored only by the local node.

See also *private resource (RMS)*, *shared resource*.

resource definition (RMS)

See *object definition (RMS)*.

resource label (RMS)

The name of the resource as displayed in a system graph.

resource state (RMS)

Current state of a resource.

RMS

See *Reliant Monitor Services (RMS)*.

RMS commands (RMS)

Commands that enable RMS resources to be administered from the command line.

RMS configuration (RMS)

A configuration made up of two or more nodes connected to shared resources. Each node has its own copy of operating system and RMS software, as well as its own applications.

RMS Wizard Kit (RMS)

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

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

RMS Wizard Tools (RMS)

A software package composed of various configuration and administration tools used to create and manage applications in an RMS configuration.

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

SAN

See *Storage Area Network*.

scalability

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

Scalable Internet Services (SIS)

The package that dynamically balances network traffic loads across cluster nodes while maintaining normal client/server sessions for each connection.

SCON

See *single console*.

script (RMS)

A shell program executed by the base monitor in response to a state transition in a resource. The script may cause the state of a resource to change.

service node (SIS)

Service nodes provide one or more TCP services (such as FTP, Telnet, and HTTP) and receive client requests forwarded by the gateway nodes.

See also *database node (SIS)*, *gateway node (SIS)*, *Scalable Internet Services (SIS)*.

SF

See *Shutdown Facility*.

shared resource

A resource, such as a disk drive, that is accessible to more than one node.

See also *private resource (RMS)*, *resource (RMS)*.

Shutdown Facility

The PRIMECLUSTER interface that manages the shutdown of cluster nodes. The SF is automatically invoked during failover operations. It also notifies other PRIMECLUSTER products of the successful completion of node shutdown so that recovery operations can begin.

Simple Network Management Protocol

A set of protocols that facilitates the exchange of information between managed network devices. The protocols are implemented by software agents residing in the devices. Each agent can read and write data in the local Management Information Base (MIB) in response to SNMP requests from other devices on the network.

See also *Management Information Base*.

single console

The workstation that acts as the single point of administration for nodes being monitored by RMS. The single console software, SCON, is run from the single console.

SIS

See *Scalable Internet Services (SIS)*.

SNMP

See *Simple Network Management Protocol*.

state

See *resource state (RMS)*.

Storage Area Network

The high-speed network that connects multiple, external storage units and storage units with multiple computers. The connections are generally fiber channels.

subapplication (RMS)

A part of the configuration template that is designed to configure one resource type for high availability. The RMS configuration may include multiple instances of each resource type. The Generic template contains subapplications for commands, application controllers, IP addresses, virtual network interfaces, local and remote file systems, volume managers, and storage managers.

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 RMS configuration file.

See also *configuration file (RMS)*.

template

See *application template (RMS)*.

type

See *object type (RMS)*.

UP (CF)

A node state that indicates that the node can communicate with other nodes in the cluster.

See also *DOWN (CF)*, *LEFTCLUSTER (CF)*, *node state (CF)*.

virtual disk

A pseudo-device that allows a portion or a combination of physical disks to be treated as a single logical disk. The virtual disk driver is inserted between the highest level of the OS logical input/output (I/O) system and the physical device driver(s), allowing all logical I/O requests to be mapped to the appropriate area on the physical disk(s).

Web-Based Admin View

A Java-based, OS-independent interface to PRIMECLUSTER management components.

See also *Cluster Admin*.

wizard (RMS)

An interactive software tool that creates a specific type of application using pretested object definitions.

Wizard Kit (RMS)

See *PCS Wizard Kit (PCS)*, *RMS Wizard Kit (RMS)*.

Wizard Tools (RMS)

See *RMS Wizard Tools (RMS)*.

Abbreviations

AC	Access Client
API	application program interface
ASCC	Adaptive Services Control Center
bm	base monitor
CCBR	Cluster Configuration Backup/Restore
CDL	Configuration Definition Language
CF	Cluster Foundation or Cluster Framework
CIM	Cluster Integrity Monitor
CIP	Cluster Interconnect Protocol
CLI	command line interface
CLM	Cluster Manager
CRM	Cluster Resource Management
DHCP	Dynamic Host Control Protocol
DLPI	Data Link Provider Interface
ELM	Enhanced Lock Manager

Abbreviations

ENS	Event Notification Services
GDS	Global Disk Services
GFS	Global File Services
GLS	Global Link Services
GUI	graphical user interface
HA	high availability
ICF	Internode Communication Facility
I/O	input/output
IPMI	Intelligent Platform Management Interface
JOIN	cluster join services module
LAN	local area network
MDS	Meta Data Server
MIB	Management Information Base
MIPC	Mesh Interprocessor Communication
MMB	Management Board (on PRIMEQUEST systems)
NIC	network interface card

NSM	Node State Monitor
OSD	operating system dependent
PAS	Parallel Application Services
PCS	PRIMECLUSTER Configuration Services
RCCU	Remote Console Control Unit
RCI	Remote Cabinet Interface
RMS	Reliant Monitor Services
SA	Shutdown Agent
SAN	Storage Area Network
SCON	single console software
SD	Shutdown Daemon
SF	Shutdown Facility
SIS	Scalable Internet Services
SNMP	Simple Network Management Protocol
VIP	Virtual Interface Provider
VNC	Virtual Network Connection

Figures

Figure 1:	Object hierarchy for initializing examples	12
Figure 2:	System graph for initializing examples—PCS	13
Figure 3:	hvdsp output for initializing examples—PCS	13
Figure 4:	Example of maintenance mode restrictions	40
Figure 5:	Scalable controller attributes	51
Figure 6:	Maintenance mode controller example	60
Figure 7:	Switching scalable parent with child in maintenance mode	61
Figure 8:	Switching scalable child with parent in maintenance mode	61
Figure 9:	Attempting to take a parent offline with the child in maintenance mode	62
Figure 10:	Attempting to take a child offline with the parent in maintenance mode	63
Figure 11:	PCS Detector Details menu	76
Figure 12:	PCS Detector Details menu for pcsdet_system	77
Figure 13:	Wizard Tools Detector Details menu	78
Figure 14:	Viewing the RMS switchlog file using a context menu . . .	83
Figure 15:	Viewing the RMS switchlog file using the Tools menu . . .	83
Figure 16:	Viewing an application log using a context menu	84
Figure 17:	RMS switchlog in tab view	85
Figure 18:	Detail of tab view showing detach button	85
Figure 19:	RMS switchlog in detached view	86
Figure 20:	Detail of detached window view showing attach button . .	86
Figure 21:	Search based on date and time range	88
Figure 22:	Search based on resource name	89
Figure 23:	Search based on severity level	90

Figures

Figure 24:	Search based on keyword	91
Figure 25:	Using the pop-up Find dialog in log viewer	92
Figure 26:	Manually executing a script from a PCS graph context menu	96
Figure 27:	PCS GUI Manual Script Execution output window	96
Figure 28:	PCS CUI Configuration Start screen	97
Figure 29:	PCS CUI Advanced Menu	98
Figure 30:	PCS CUI Manual Script Execution Tree—selecting an object	98
Figure 31:	PCS CUI Manual Script Execution Tree—selecting a script	99
Figure 32:	PCS CUI Manual Script Execution Tree—viewing script results 99	
Figure 33:	Wizard Tools Main configuration menu	100
Figure 34:	Wizard Tools ManualExecution: Application selection menu	100
Figure 35:	Wizard Tools ManualExecution: Resource selection menu	101
Figure 36:	Wizard Tools ManualExecution: Script selection menu . . .	101

Tables

Table 1: Dependence of scalable controller states on child applications	52
Table 2: Log files	68
Table 3: Base monitor log levels	71
Table 4: RMS severity level description	90
Table 5: Solaris errno error numbers and their meanings	273
Table 6: Linux errno error numbers and their meanings	279
Table 7: States reported by detectors for RMS objects	285
Table 8: Additional states that may be displayed for RMS objects . . .	285
Table 9: StateDetails values for RMS objects	287

Index

A

- administrator prerequisites 1
- Affiliation attribute 299
- alerts
 - log viewer 90
- Alternatelp attribute 291
- andOp objects
 - defined* 289
- application logs
 - searching text 92
 - viewing 84
- applications
 - going offline 22
 - log files 72
 - overriding AutoStartUp 309
 - switching 35
 - switching to SysNode 317
 - viewing log files 83
 - viewing logs 84
- ASCC product family
 - administrator prerequisites 1
- attach button
 - graph window 86
- attributes
 - Affiliation 299
 - Alternatelp 291
 - AutoRecover 291
 - AutoRecoverCleanup 292
 - AutoStartUp 292
 - AutoSwitchOver 292
 - Class 299
 - ClusterExclusive 292
 - Comment 299
 - ControlledSwitch 293
 - DetectorStartScript 299
 - ENV object 289
 - FaultScript 293
 - for andOp objects 289
 - for gController objects 289
 - for gResource objects 289
 - for orOp objects 289
 - for SysNode objects 290
 - for userApplication objects 290
 - Halt 293
 - HostName 300
 - I_List 294
 - LastDetectorReport 300
 - LieOffline 294
 - MaxControllers 300
 - MonitorOnly 294
 - NoDisplay 300
 - NullDetector 300
 - OfflineDoneScript 300
 - OfflineScript 294
 - OnlinePriority 294
 - OnlineScript 295
 - PartialCluster 295
 - PersistentFault 296
 - PostOfflineScript 296
 - PostOnlineScript 296
 - PreCheckScript 301
 - PreOfflineScript 296
 - PreOnlineScript 296
 - PreserveState 296
 - PriorityList 297
 - Resource 297
 - rName 301
 - ScriptTimeout 297
 - ShutdownPriority 297
 - SplitRequest 301
 - StandbyCapable 298
 - StateDetails 301
 - WarningScript 298
- AutoRecover attribute
 - defined* 291
 - fault processing 29
- AutoRecoverCleanup attribute 292
- AutoStartUp attribute
 - defined* 292
 - overriding 309
- AutoSwitchOver attribute
 - defined* 292

- fault processing 26, 27
- B**
- base monitor
 - cluster monitoring 307
 - communication port 306
 - debug messages 67
 - heartbeat 104
 - locking process in memory 310
 - log file 69
 - log levels 71
 - messages 69
 - object states 285
 - setting log levels 70
 - stack tracing 71
 - switchlog 72
- bmlog description 69
- buttons
 - attach, graph window 86
 - detach, graph tab view 85
 - Filter, log viewer 87
- C**
- case-insensitive
 - searching logs 91
- case-sensitive
 - searching logs 92
- Caution
 - defined* 7
 - Do not explicitly set RMS
 - environment variables in the user environment 303
 - If the HV_AUTOSTARTUP_IGNORE environment variable is used, ensure that it is correctly defined on all cluster nodes and that it is always kept up-to-date 304
 - The pcsloglev setting persists across configuration activations, RMS restarts, and node reboots. 77
- CF
 - LEFTCLUSTER 33
 - required for ELM 307
 - CF commands
 - cfconfig 319
 - cfrecon 319
 - cftool 319
 - rcqconfig 320
 - cfset command 319
 - changing
 - environment variables 303
 - CIP
 - cipconfig commands 320
 - ciptool command 320
 - cip.cf file 320
 - Class attribute 299
 - clbackuprdb command 321
 - clearing
 - hung nodes 317
 - resource faults 317
 - clgettree command 321
 - CLI *see* RMS commands
 - clinitreaset command 321
 - clrestorerdb command 321
 - clsetparam command 321
 - clsetup command 321
 - clstarttrsc command 321
 - clstoprsc command 321
 - cluster 1
 - monitoring by RMS 307
 - Cluster Admin
 - searching log text 92
 - switchlog 83
 - viewing log files 83
 - Cluster Admin GUI
 - administrator prerequisites 1
 - Cluster Foundation 43
 - Cluster Foundation *see* CF
 - cluster nodes *see* nodes
 - cluster_uninstall command 326
 - ClusterExclusive attribute
 - defined* 292
 - commands
 - hvdump 105
 - commands *see* RMS commands

- Comment attribute 299
- communication port
 - base monitor 306
- CONFIG.rms
 - default RMS startup file 310
- configuration tools log files 72
- configurations
 - displaying 316
- configuring
 - ssh access between nodes 326
- console error messages 209
- context menus
 - equivalence 84
 - starting script 95
 - viewing switchlog 83
- ControlledSwitch attribute 293
- controller objects
 - defined* 289
- critical errors
 - log viewer 90
- cron jobs
 - hvlogclean 93
 - hvlogcontrol 94
- D**
- Deact state 285
- debug level, wizards 79
- debug messages 65
 - base monitor 67
 - log directory 67
 - log viewer 90
 - scripts, PCS and Wizards Tools 73
- debug reporting, wizards 78
- defining
 - shutdown timeout 308
- deleting
 - log files 316
 - log files by age 308
 - log files by size 310
- detach button
 - graph tab view 85
- detector
 - trace, set automatically 105
- detectors
 - fault situations 20, 25
 - illegal 11
 - log files 73, 74
 - names 74
 - setting log levels with hvutil -L 75, 76
 - setting log levels with Wizard Tools 78
 - switchlog file 72
- DetectorStartScript attribute 299
- dialogs
 - find, log viewer 92
- directed switch requests 35
- directories
 - RMS, specifying root 308
- displaying
 - current RMS configuration 316
 - manual pages 319
- documentation
 - online 319
 - related 2
- double faults 27
 - defined* 293
 - and Halt attribute 27, 293
- E**
- ELM
 - defined* 43
 - enabling and disabling 307
- emergency
 - log viewer 90
- Enhanced Lock Manager *see* ELM
- ENV object 289
- environment variables
 - HV_APPLICATION 55, 313
 - HV_APPLICATION_STATE_CHA
NGE_INFO, scalable controller
script 55
 - HV_AUTORECOVER 313
 - HV_AUTORECOVER script
AutoRecover attempt flag 55
 - HV_AUTOSTART_WAIT 305
 - HV_AUTOSTARTUP 309

- HV_AUTOSTARTUP_IGNORE 304
- HV_CHECKSUM_INTERVAL 305
- HV_COM_PORT 306
- HV_CONNECT_TIMEOUT 309
- HV_FORCED_REQUEST 313
- HV_FORCED_REQUEST script forced request flag 55
- HV_HOST_STATE_CHANGE_IN FO, scalable controller script 55
- HV_LAST_DET_REPORT 313
- HV_LAST_DET_REPORT, script last detector report 55
- HV_LOG_ACTION 309
- HV_LOG_ACTION_THRESHOLD 306
- HV_LOG_WARN_THRESHOLD 306
- HV_MAXPROC 310
- HV_MLOCKALL 310
- HV_NODENAME 313
- HV_NODENAME, script current object name 56
- HV_OFFLINE_REASON 313
- HV_OFFLINE_REASON, script reason for offline processing 56
- HV_RCSTART, *defined* 310
- HV_REALTIME_PRIORITY 310
- HV_REQUESTING_CONTROLLED R script forced request flag 313
- HV_SCALABLE_CONTROLLER, scalable controller script 314
- HV_SCALABLE_INFO, scalable controller script 314
- HV_SCRIPT_TYPE 314
- HV_SCRIPT_TYPE, script type 56
- HV_SCRIPTS_DEBUG 311
- HV_SYSLOG_USE 311
- HV_USE_ELM 307
- NODE_SCRIPTS_TIME_OUT 314
- NODE_SCRIPTS_TIME_OUT, script timeout value for current object 56
- RELIANT_HOSTNAME 311
- RELIANT_INITSCRIPT 312
- RELIANT_LOG_LIFE 307
- RELIANT_LOG_PATH 67, 308
- RELIANT_PATH 308
- RELIANT_SHUT_MIN_WAIT 308
- RELIANT_STARTUP_PATH 312
- SCRIPTS_TIME_OUT 312
- ENVL object 289
- error levels
 - log viewer 90
- error messages 65
 - base monitor 67
 - console 209
 - fatal 191
 - notice 267
 - switchlog 107
 - switchlog, script timeout 35
 - warning 233
- errors
 - at initialization 21
 - during offline processing 25
 - in offline state 28
 - reaction to 26
- /etc/inittab 316
- expert operations
 - disabling ELM 307
- F**
- failover
 - defined* 42
- fatal error messages 191
- fatal errors
 - log viewer 90
- fault script
 - RMS reaction to fault 26
- Faulted state
 - defined* 285
 - clearing 32

- faults
 - SysNode 33
- FaultScript attribute 293
- file systems
 - filling up 79
 - warning threshold 306
- Filter button
 - log viewer 87
- filters
 - log viewer 87
- fjsvwvbs command 325, 326
- fjswvcnf command 325
- forcing
 - online requests 33
- G**
- GDS subapplication
 - WarningScript 298
- graphs
 - detaching view 86
 - reinitializing 317
 - tabbed view 85
- gResource objects
 - defined* 289
- H**
- Halt attribute
 - defined* 293
 - and double faults 27
- heartbeat, base monitor 104
- heartbeats
 - and SysNode faults 33
 - cluster monitoring 42
 - mode 307
 - UDP, and ELM 307
- high availability 1
- HostName attribute
 - defined* 300
 - andOp objects 289
- HTML documentation 326
- HV_APPLICATION 55, 313
- HV_APPLICATION_STATE_CHANG
E_INFO 55
- HV_AUTORECOVER 55, 313
- HV_AUTOSTART_WAIT 305
- HV_AUTOSTARTUP
 - defined* 309
 - AutoStartUp attribute 292
- HV_AUTOSTARTUP_IGNORE 304
- HV_AUTOSTARTUP_WAIT
 - and PartialCluster attribute 295
- HV_CHECKSUM_INTERVAL 305
- HV_COM_PORT 306
- HV_CONNECT_TIMEOUT 309
- HV_FORCED_REQUEST 55, 313
- HV_HOST_STATE_CHANGE_INFO
55
- HV_LAST_DET_REPORT 55, 313
- HV_LOG_ACTION 93, 309
- HV_LOG_ACTION_THRESHOLD
93, 306
- HV_LOG_WARN_THRESHOLD 93,
306
- HV_MAX_HVDISP_FILE_SIZE 310
- HV_MAXPROC 310
- HV_MLOCKALL 310
- HV_NODENAME 56, 313
- HV_OFFLINE_REASON 56, 313
- HV_RCSTART
 - defined* 310
- HV_REALTIME_PRIORITY 310
- HV_REQUESTING_CONTROLLER
313
- HV_SCALABLE_CONTROLLER
314
- HV_SCALABLE_INFO 314
- HV_SCRIPT_TYPE 56, 314
- HV_SCRIPTS_DEBUG 311
- HV_SYSLOG_USE 311
- HV_USE_ELM 307
- hvassert command
 - defined* 315
- hvattr command
 - defined* 315
- hvcn command
 - defined* 315
 - h option 307
- hvconfig command

Index

- defined* 316
- hvdisp command
 - displaying environment variables 304
 - file size 310
 - no display 300
- hvdist command
 - defined* 316
- hvdump command 105
 - defined* 316
- hvgdmake command
 - defined* 316
- hvmlogclean command 93
 - defined* 316
- hvmlogcontrol command 93
 - log file cleanup 306
- hvmrclev command
 - defined* 316
- hvmreset command
 - defined* 317
- hvmsetenv command
 - defined* 317
 - HV_AUTOSTARTUP 309
 - HV_RCSTART 310
- hvmshut command
 - defined* 317
 - defining timeout 308
- hvmswitch command
 - defined* 317
 - f option 33
- hvmutil
 - l base monitor log level 70
 - L, setting detector log levels 75
- hvmutil command
 - defined* 317
 - c option 32, 33
- I**
- I_List attribute 294
- Inconsistent state
 - defined* 285
- informational messages
 - log viewer 90
- initialization
 - error during 21
 - initial object state 10
 - objects 11
 - script, specifying 312
 - Unknown state 11
- intended state
 - maintenance mode 286
- K**
- kernel
 - ELM locks 43
- L**
- LastDetectorReport attribute 300
- LEFTCLUSTER
 - CF event, SysNode fault 33
- LieOffline attribute 294
- locking
 - base monitor in memory 310
- locks, managed by ELM 43
- log file cleanup 93
- log files
 - applications 72
 - base monitor 69
 - detectors 73, 74
 - PCS and Wizard Tools 72
 - PCS detectors 74
 - searching 92
 - specify directory 308
 - switchlog 35, 68
 - time of preservation 308
 - viewing 83, 84
 - Wizard Tools detectors 74
- log levels
 - controlling 70
 - detectors, setting in Wizard Tools 78
 - detectors, setting with hvutil -L 75, 76
 - detectors, setting with Wizard Tools 78
- log messages, wizards 72
- log viewer
 - described 83

- error levels 90
- filters 87
- M**
- MA commands
 - clrcumonctl 325
 - clrcimonctl 324, 325
- maintenance mode
 - Maintenance state 286
 - setting 317
- manual pages
 - displaying 319
 - RMS commands 315
- manual script execution
 - PCS CUI 97
 - PCS GUI 95
 - Wizard Tools 95
- MaxControllers attribute 300
- messages
 - base monitor 69
 - between RMS objects 10
 - bmlog 69
 - debug 67
 - error 67
 - generic detector log 68
 - troubleshooting RMS 101
 - wizards 72
- messages, error
 - console 209
 - fatal 191
 - notice 267
 - switchlog 107
 - warning 233
- MonitorOnly attribute
 - defined* 294
- N**
- NODE_SCRIPTS_TIME_OUT 56, 314
- nodes
 - detector timeout for remote 309
 - ignore at startup 304
 - wait to report online 305
- NoDisplay attribute 300
- non-fatal errors
 - log viewer 90
- notice error messages 267
- notices
 - log viewer 90
- NullDetector attribute 300
- O**
- object types
 - andOp 289
 - ENV 289
 - ENVL 289
 - gController 289
 - gResource 289
 - orOp 289
 - SysNode 290
 - userApplication 290
- offline processing
 - defined* 22
 - fault situations 25
 - requests 22
- Offline state
 - defined* 285
- OfflineDoneScript attribute 300
- OfflineFault state 285
- OfflineScript attribute 294
- online manual pages 319
- Online state
 - defined* 285
- OnlinePriority attribute 294
- OnlineScript attribute 295
- operating system
 - administrator prerequisites 1
- operator
 - intervention 35
- P**
- PartialCluster attribute
 - defined* 295
 - and HV_AUTOSTART_WAIT 305
- PAS commands
 - clmtest 320
 - mipcstat 320
- PCS

Index

- administrator prerequisites 1
- script debug messages 73
- setting detector log levels 76
- PCS detectors
 - log file location 74
- PCS log files 72
- pcsloglev command 77
- PersistentFault attribute 296
- physical disks
 - state at initialization 21
- PostOfflineScript attribute 296
- PostOnlineScript attribute 296
- PreCheckScript attribute 301
- PreOffline processing 23
- PreOfflineScript attribute 296
- PreOnlineScript attribute 296
- PreserveState attribute
 - defined* 296
 - effect on fault processing 26, 27, 28
- PRIMECLUSTER product family
 - administrator prerequisites 1
- priority switch
 - request 35
- PriorityList attribute
 - defined* 297
- R**
- rcsd command 323
- rcsd.cfg file 323
- recovery timeout
 - UDP heartbeats 42, 307
- related documentation 2
- RELIANT_HOSTNAME 311
- RELIANT_INITSCRIPT 312
- RELIANT_LOG_LIFE 307
- RELIANT_LOG_PATH 67, 308
- RELIANT_PATH 308
- RELIANT_SHUT_MIN_WAIT 308
- RELIANT_STARTUP_PATH 312
- requests
 - in RMS messages 10
 - offline processing 22
- Resource attribute
 - defined* 297
 - controller objects 289
- resources
 - clearing faulted 317
- right pane
 - log messages 85
- right-click, mouse
 - starting script 95
- rKind attribute
 - gResource objects 289, 301
- RMS
 - severity levels 90
- RMS commands
 - see also* individual commands
 - basic list* 315
 - hvassert 315, 322
 - hvattrib 315
 - hvcmm 315, 322
 - hvconfig 316, 322
 - hvdisp 316, 322
 - hvdist 316, 322
 - hvdump 316, 322
 - hvenv.local 323
 - hvgdmake 316, 322
 - hvlogclean 93, 316, 322
 - hvlogcontrol 93
 - hvrclv 316
 - hvreset 317, 322
 - hvsetenv 317, 322
 - hvshut 317, 322
 - hvswitch 33, 317, 322
 - hvthrottle 317, 323
 - hvutil 32, 33, 317, 323
- RMS Reference Guide 87
- rName attribute
 - gResource objects 289, 301
- rolling upgrades
 - disabling ELM 43
- S**
- SA_blade.cfg file 323
- SA_rccu.cfg file 323
- SA_rps.cfg file 323
- SA_rsb.cfg file 324

-
- SA_SATxscf2.cfg file 324
 - SA_SATxscf.cfg file 324
 - SA_scon.cfg file 323, 324
 - SA_snmp.cfg file 324
 - SA_sspint.cfg file 324
 - SA_sunF.cfg file 324
 - SA_wtinps.cfg file 324
 - script top-of-tree 55
 - scripts
 - PCS and Wizard Tools, debugging 73
 - timeout 312
 - top-of-tree 313
 - SCRIPTS_TIME_OUT 312
 - ScriptTimeout attribute
 - defined* 297
 - sdtool command 323
 - searching logs
 - in log viewer 92
 - sending
 - clear-fault request 32, 33
 - setting
 - log levels with hvutil -L 75, 76
 - log levels with Wizard Tools 78
 - severity levels *see* error levels
 - ShutdownPriority attribute
 - defined* 297
 - SIS
 - dtcpadmin command 325
 - dtcpd command 325
 - dtcpdbg command 325
 - software monitor
 - function 1
 - SplitRequest attribute 301
 - Standby state
 - defined* 285
 - StandbyCapable attribute 298
 - starting
 - RMS, and heartbeat recovery timeout 43
 - StateDetails attribute
 - defined* 301
 - displaying 287
 - states
 - definitions* 285
 - Deact 285
 - details, displaying 287
 - Faulted 285
 - Inconsistent, *defined* 285
 - Maintenance 286
 - Offline 285
 - OfflineFault 285
 - Online 285
 - Standby 285
 - Unknown 285
 - Wait 286
 - Warning 286
 - status
 - communication, CF 319
 - SIS 325
 - Web-Based Admin View 326
 - storage managers
 - administrator prerequisites 1
 - strace, Linux trace tool 105
 - switching
 - defined* 35
 - applications 317
 - switchlog
 - file 67, 68
 - script timeout error messages 35
 - viewing 83
 - switchlog error messages 107
 - SysNode objects
 - defined* 290
 - fault 33
 - initializing 11
 - switching application to 317
- T**
- tabbed view
 - graphs 85
 - log viewer 85
 - timeouts
 - heartbeat recovery 42
 - trace
 - detector, set automatically 105
 - truss, Solaris trace tool 105
-

Index

U

- UDP protocol
 - RMS heartbeats 42, 307
- Unknown state
 - defined* 285
 - exiting 10
 - initial state 11
 - RMS initialization 10
- userApplication objects
 - defined* 290
 - independent initialization 11

V

- /var/opt/SMAWRrms/log/ 83
- viewing
 - application logs 84
 - graphs, attaching and detaching 85, 86
 - log viewer 85
- volume managers
 - administrator prerequisites 1

W

- Wait state
 - defined* 286
 - clearing faulted resources 317
 - clearing hung nodes 317
 - script timeout limit for node kill 35
- warning error messages 233
- Warning state
 - defined* 286
 - WarningScript 298
- warnings
 - log viewer 90
- WarningScript
 - GDS subapplication 298
- WarningScript attribute 298
- Wizard Tools
 - administrator prerequisites 1
 - script debug messages 73
 - setting detector log levels 78
- Wizard Tools detectors
 - log file location 74
 - setting log level 78

- Wizard Tools log files 72
- wizards
 - debug level 79
 - debug reporting 78
- wvCntl command 325
- wvconf command 325, 326
- wvGetparam command 325
- wvSetparam command 325
- wvstat command 326