

# **Managing HP Serviceguard for Linux, Sixth Edition**



**Manufacturing Part Number : B9903-90050**

**August 2006**

---

## Legal Notices

The information in this document is subject to change without notice.

*Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.* Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

**Warranty.** A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

**Restricted Rights Legend.** Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Company  
19420 Homestead Road  
Cupertino, California 95014 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

### Copyright Notices

© Copyright 2001–2006 Hewlett-Packard Development Company, L.P.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under copyright laws.

**Trademark Notices**

HP Serviceguard® is a registered trademark of Hewlett-Packard Company, and is protected by copyright.

NIS™ is a trademark of Sun Microsystems, Inc.

UNIX® is a registered trademark of The Open Group.

Linux® is a registered trademark of Linus Torvalds.

Red Hat® is a registered trademark of Red Hat Software, Inc.



**1. Serviceguard for Linux at a Glance**

What is Serviceguard for Linux? .....	18
Failover .....	20
Viewing Serviceguard Clusters .....	22
Configuration Roadmap .....	23

**2. Understanding Hardware Configurations for Serviceguard for Linux**

Redundancy of Cluster Components .....	26
Redundant Network Components .....	27
Redundant Ethernet Configuration .....	27
Redundant Disk Storage .....	28
Supported Disk Interfaces .....	28
Disk Monitoring .....	28
Sample Disk Configurations .....	29
Redundant Power Supplies .....	30

**3. Understanding Serviceguard Software Components**

Serviceguard Architecture .....	32
Serviceguard Daemons .....	32
How the Cluster Manager Works .....	38
Configuration of the Cluster .....	38
Heartbeat Messages .....	38
Manual Startup of Entire Cluster .....	39
Automatic Cluster Startup .....	40
Dynamic Cluster Re-formation .....	40
Cluster Quorum to Prevent Split-Brain Syndrome .....	41
Cluster Lock .....	41
Use of a Lock LUN as the Cluster Lock .....	41
Use of the Quorum Server as a Cluster Lock .....	42
Types of Quorum Server Configuration .....	43
How the Package Manager Works .....	45
Configuring Packages .....	46
Deciding When and Where to Run and Halt Packages .....	46
Package Switching .....	46
Failover Policy .....	49
Failback Policy .....	52
Choosing Package Failover Behavior .....	55
How Package Control Scripts Work .....	57

---

# Contents

What Makes a Package Run? .....	57
Before the Control Script Starts .....	59
During Run Script Execution .....	59
Normal and Abnormal Exits from the Run Script .....	61
Service Startup with cmrunserv .....	61
Disk Monitor Services .....	62
While Services are Running .....	62
When a Service or Subnet Fails .....	63
When a Package is Halted with a Command .....	63
During Halt Script Execution .....	64
Normal and Abnormal Exits from the Halt Script .....	65
How the Network Manager Works .....	68
Stationary and Relocatable IP Addresses .....	68
Adding and Deleting Relocatable IP Addresses .....	69
Bonding of LAN Interfaces .....	70
Bonding for Load Balancing .....	73
Remote Switching .....	74
ARP Messages after Switching .....	74
Volume Managers for Data Storage .....	75
Examples of Storage on Smart Arrays .....	76
Multipathing and LVM .....	79
Monitoring Disks .....	80
More Information on LVM .....	80
Responses to Failures .....	81
Transfer of Control (TOC) When a Node Fails .....	81
Responses to Hardware Failures .....	82
Responses to Package and Service Failures .....	83

## 4. Planning and Documenting an HA Cluster

General Planning .....	87
Serviceguard Memory Requirements .....	88
Planning for Expansion .....	88
Hardware Planning .....	89
SPU Information .....	90
LAN Information .....	90
Shared Storage .....	92

Disk I/O Information .....	93
Hardware Configuration Worksheet .....	95
Power Supply Planning .....	96
Power Supply Configuration Worksheet .....	96
Quorum Server Planning .....	97
Quorum Server Worksheet .....	97
Volume Manager Planning .....	98
Volume Groups and Physical Volume Worksheet .....	98
Cluster Configuration Planning .....	99
Quorum Server Information .....	100
Lock LUN Information .....	101
Cluster Configuration Parameters .....	101
Cluster Configuration Worksheet .....	106
Package Configuration Planning .....	107
Logical Volume and File System Planning .....	107
Planning for Expansion .....	109
Choosing Switching and Failover Behavior .....	109
Package Configuration File Parameters .....	109
Package Control Script Variables .....	114
Package Configuration Worksheet .....	118

## **5. Building an HA Cluster Configuration**

Preparing Your Systems .....	122
Understanding the Location of Serviceguard Files .....	122
Enabling Serviceguard Command Access .....	123
Editing Security Files .....	124
Username Validation .....	126
Access Roles .....	127
Setting Access Controls for a Configured Clusters .....	130
Setting up the Quorum Server .....	133
Installing the Quorum Server .....	134
Running the Quorum Server .....	135
Creating a Package for the Quorum Server .....	136
Setting up the Lock LUN .....	138
Implementing Channel Bonding (Red Hat) .....	140
Sample Configuration .....	141
Restarting Networking .....	142
Viewing the Configuration .....	143

---

# Contents

Implementing Channel Bonding (SLES9 and SLES10) .....	144
Restarting Networking .....	145
Creating the Logical Volume Infrastructure .....	147
Displaying Disk Information .....	149
Creating Partitions .....	149
Enabling VG Activation Protection .....	152
Building Volume Groups: Example for Smart Array Cluster Storage (MSA 500 Series)	
153	
Building Volume Groups and Logical Volumes .....	156
Distributing the Shared Configuration to all Nodes .....	157
Testing the Shared Configuration .....	158
Storing Volume Group Configuration Data .....	160
Setting up Disk Monitoring .....	161
Configuring the Cluster .....	162
Using Serviceguard Manager to Configure the Cluster .....	163
Specifying a Quorum Server .....	163
Specifying a Lock LUN .....	164
Cluster Configuration Template File .....	164
Specifying Maximum Number of Configured Packages .....	168
Modifying Cluster Timing Parameters .....	169
Adding or Removing Nodes While the Cluster is Running .....	169
Verifying the Cluster Configuration .....	170
Cluster Lock Configuration Messages .....	171
Distributing the Binary Configuration File .....	171
Managing the Running Cluster .....	172
Checking Cluster Operation with Serviceguard Manager .....	172
Checking Cluster Operation with Serviceguard Commands .....	172
Setting up Autostart Features .....	175
Changing the System Message .....	176
Managing a Single-Node Cluster .....	176
Deleting the Cluster Configuration .....	177

## 6. Configuring Packages and Their Services

Using Serviceguard Manager to Configure a Package .....	180
Creating the Package Configuration .....	181
Creating the Package Configuration File .....	181



Writing the Package Control Script .....	188
Customizing the Package Control Script .....	189
Optimizing for Large Numbers of Storage Units .....	190
Configuring Disk Monitoring Services .....	190
Package Control Script Template File .....	191
Adding Customer Defined Functions to the Package Control Script .....	202
Adding or Removing Packages on a Running Cluster .....	203
Verifying the Package Configuration .....	204
Applying and Distributing the Configuration .....	205
Copying Package Control Scripts with Linux commands .....	205
Testing Cluster and Package Operation .....	205
Creating a Disk Monitor Configuration .....	206
Configuring All Disks for Monitoring .....	207
Configuring Disks on a Single Node for Monitoring .....	209

## **7. Cluster and Package Maintenance**

Reviewing Cluster and Package States with the <code>cmviewcl</code> Command .....	212
Types of Cluster and Package States .....	213
Examples of Cluster and Package States .....	217
Using Serviceguard Manager .....	224
How Serviceguard Manager Works .....	224
Running Serviceguard Manager with a Command .....	225
Starting with a Specific Cluster .....	226
Starting Serviceguard Manager without a Specific Cluster .....	228
Connecting to an Object Manager .....	230
Opening a Saved File with Cluster Data .....	231
Viewing Cluster Data .....	232
Obtaining Help .....	235
Managing Cluster Objects .....	236
Viewing Status of Monitored Disks .....	240
Managing the Cluster and Nodes .....	242
Starting the Cluster When all Nodes are Down .....	243
Adding Previously Configured Nodes to a Running Cluster .....	244
Removing Nodes from Operation in a Running Cluster .....	245
Halting the Entire Cluster .....	246
Reconfiguring a Halted Cluster .....	247
Automatically Restarting the Cluster .....	247
Reconfiguring a Running Cluster .....	248

---

# Contents

Adding Nodes to the Configuration While the Cluster is Running .....	249
Deleting Nodes from the Configuration While the Cluster is Running .....	250
Managing Packages and Services .....	251
Starting a Package .....	251
Halting a Package .....	252
Moving a Package .....	253
Reconfiguring a Package on a Halted Cluster .....	254
Reconfiguring a Package on a Running Cluster .....	255
Adding a Package to a Running Cluster .....	256
Deleting a Package from a Running Cluster .....	256
Changing Package Switching Behavior .....	257
Resetting the Service Restart Counter .....	257
Allowable Package States During Reconfiguration .....	258
Responding to Cluster Events .....	260
Single-Node Operation .....	261
Removing Serviceguard from a System .....	262

## 8. Troubleshooting Your Cluster

Testing Cluster Operation .....	264
Testing the Package Manager .....	264
Testing the Cluster Manager .....	265
Testing the Network Manager .....	266
Monitoring Hardware .....	267
Replacing Disks .....	268
Replacing a Faulty Mechanism in a Disk Array .....	268
Replacement of LAN Cards .....	269
Replacing a Failed Quorum Server System .....	271
Troubleshooting Approaches .....	273
Reviewing Package IP Addresses .....	274
Reviewing the System Log File .....	275
Reviewing Object Manager Log Files .....	277
Reviewing Configuration Files .....	277
Reviewing the Package Control Script .....	277
Using the cmquerycl and cmcheckconf Commands .....	278
Reviewing the LAN Configuration .....	278
Solving Problems .....	279

Serviceguard Command Hangs .....	279
Cluster Re-formations .....	280
System Administration Errors .....	280
Package Movement Errors .....	283
Node and Network Failures .....	283
Quorum Server Messages.....	284
Lock LUN Messages .....	284

## **A. Serviceguard Commands**

## **B. Designing Highly Available Cluster Applications**

Automating Application Operation .....	298
Insulate Users from Outages .....	298
Define Application Startup and Shutdown .....	299
Controlling the Speed of Application Failover .....	300
Replicate Non-Data File Systems .....	300
Evaluate the Use of a Journaled Filesystem (JFS).....	301
Minimize Data Loss .....	301
Use Restartable Transactions .....	302
Use Checkpoints .....	303
Design for Multiple Servers .....	304
Design for Replicated Data Sites .....	304
Designing Applications to Run on Multiple Systems .....	305
Avoid Node Specific Information .....	305
Avoid Using SPU IDs or MAC Addresses .....	307
Assign Unique Names to Applications .....	308
Use uname(2) With Care .....	309
Bind to a Fixed Port .....	310
Bind to Relocatable IP Addresses .....	310
Give Each Application its Own Volume Group .....	312
Use Multiple Destinations for SNA Applications .....	312
Avoid File Locking .....	313
Restoring Client Connections .....	314
Handling Application Failures .....	316
Create Applications to be Failure Tolerant .....	316
Be Able to Monitor Applications .....	317
Minimizing Planned Downtime .....	318
Reducing Time Needed for Application Upgrades and Patches .....	319

---

## Contents

Providing Online Application Reconfiguration .....	320
Documenting Maintenance Operations .....	320

### **C. Integrating HA Applications with Serviceguard**

Checklist for Integrating HA Applications .....	322
Defining Baseline Application Behavior on a Single System .....	322
Integrating HA Applications in Multiple Systems .....	323
Testing the Cluster .....	324

### **D. Blank Planning Worksheets**

Hardware Worksheet .....	326
Power Supply Worksheet .....	327
Quorum Server Worksheet .....	328
Volume Group and Physical Volume Worksheet .....	329
Cluster Configuration Worksheet .....	330
Package Configuration Worksheet .....	332
Package Control Script Worksheet .....	333

<b>Index .....</b>	<b>335</b>
--------------------	------------

---

## Printing History

**Table 1**

<b>Printing Date</b>	<b>Part Number</b>	<b>Edition</b>
November 2001	B9903-90005	First
November 2002	B9903-90012	First
December 2002	B9903-90012	Second
November 2003	B9903-90033	Third
February 2005	B9903-90043	Fourth
June 2005	B9903-90046	Fifth
August 2006	B9903-90050	Sixth

The last printing date and part number indicate the current edition, which applies to the A.11.16 version of HP Serviceguard for Linux.

The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The part number is revised when extensive technical changes are incorporated.

New editions of this manual will incorporate all material updated since the previous edition.

**HP Printing Division:**

*Business Critical Computing  
Hewlett-Packard Co.  
19111 Pruneridge Ave.  
Cupertino, CA 95014*



---

## Preface

This guide describes how to configure and manage Serviceguard for Linux on HP ProLiant and HP Integrity servers under the Linux operating system. It is intended for experienced Linux system administrators. (For Linux system administration tasks that are not specific to Serviceguard, use the system administration documentation and manpages for your distribution of Linux.)

The contents are as follows:

- Chapter 1, “Serviceguard for Linux at a Glance,” describes a Serviceguard cluster and provides a roadmap for using this guide.
- Chapter 2, “Understanding Hardware Configurations for Serviceguard for Linux,” provides a general view of the hardware configurations used by Serviceguard.
- Chapter 3, “Understanding Serviceguard Software Components,” describes the software components of Serviceguard and shows how they function within the Linux operating system.
- Chapter 4, “Planning and Documenting an HA Cluster,” steps through the planning process and provides a set of worksheets for organizing information about the cluster.
- Chapter 5, “Building an HA Cluster Configuration,” describes the creation of the cluster configuration.
- Chapter 6, “Configuring Packages and Their Services,” describes the creation of high availability packages and the control scripts associated with them.
- Chapter 7, “Cluster and Package Maintenance,” presents the basic cluster administration tasks.
- Chapter 8, “Troubleshooting Your Cluster,” explains cluster testing and troubleshooting strategies.
- Appendix A, “Serviceguard Commands,” lists the commands used by Serviceguard for Linux and reprints summary information from each man page.
- Appendix B, “Designing Highly Available Cluster Applications,” gives guidelines for creating cluster-aware applications that provide optimal performance in a Serviceguard environment.

- Appendix C, “Integrating HA Applications with Serviceguard,” presents suggestions for integrating your existing applications with Serviceguard for Linux.
- Appendix D, “Blank Planning Worksheets,” contains a set of empty worksheets for preparing a Serviceguard configuration.

## Related Publications

The following documents contain additional useful information:

- *HP Serviceguard for Linux Version A.11.16 Release Notes, Third Edition (B9903-90051)*
- *Serviceguard Manager Version A.05.00 Release Notes (B8325-90056)*
- *HP Serviceguard Quorum Server Version A.02.00 Release Notes, Fourth Edition (HP Part Number B8467-90026)*
- *Clusters for High Availability: a Primer of HP Solutions*. Second Edition. HP Press, 2001 (ISBN 0-13-089355-2)

Use the following URL to access HP’s high availability documentation web page:

**<http://docs.hp.com/hpux/ha>**

Information about supported configurations is in the *HP Serviceguard for Linux Order and Configuration Guide*. For updated information on supported hardware and Linux distributions refer to the *HP Serviceguard for Linux Certification Matrix*. Both documents are available at:

**<http://www.hp.com/info/sglx>**

**Problem Reporting** If you have any problems with the software or documentation, please contact your local Hewlett-Packard Sales Office or Customer Service Center.



# 1      **Serviceguard for Linux at a Glance**

This chapter introduces Serviceguard for Linux and shows where to find different kinds of information in this book. The following topics are presented:

- What is Serviceguard for Linux
- Viewing Serviceguard Clusters
- Configuration Roadmap

If you are ready to start setting up Serviceguard clusters, skip ahead to Chapter 4, “Planning and Documenting an HA Cluster.” Specific steps for setup are given in Chapter 5, “Building an HA Cluster Configuration.”

---

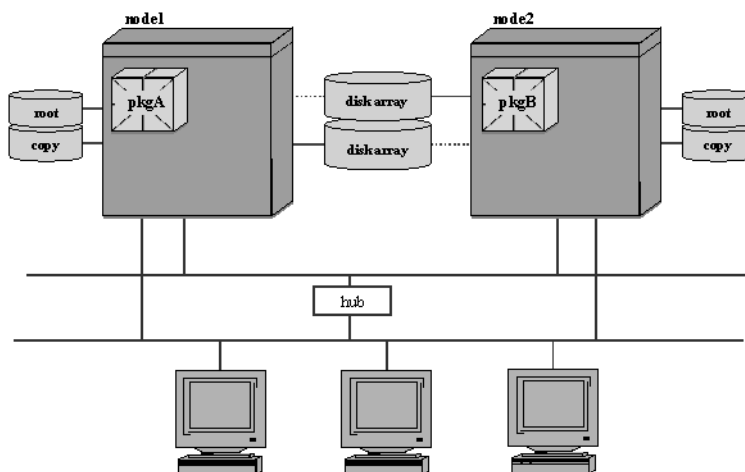
## What is Serviceguard for Linux?

**Serviceguard for Linux** allows you to create high availability clusters of HP ProLiant and HP Integrity servers. A **high availability** computer system allows application services to continue in spite of a hardware or software failure. Highly available systems protect users from software failures as well as from failure of a system processing unit (SPU), disk, or local area network (LAN) component. In the event that one component fails, the redundant component takes over. Serviceguard and other high availability subsystems coordinate the transfer between components.

A Serviceguard **cluster** is a networked grouping of HP ProLiant and HP Integrity servers (host systems known as **nodes**) having sufficient redundancy of software and hardware that a **single point of failure** will not significantly disrupt service. Application services (individual Linux processes) are grouped together in **packages**; in the event of a single service, node, network, or other resource failure, Serviceguard can automatically transfer control of the package to another node within the cluster, allowing services to remain available with minimal interruption.

Figure 1-1 Conceptual representation of a Serviceguard cluster configuration with two nodes.

**Figure 1-1** Typical Cluster Configuration



In the figure, node 1 (one of two SPU's) is running package A, and node 2 is running package B. Each package has a separate group of disks associated with it, containing data needed by the package's applications, and a copy of the data. Note that both nodes are physically connected to disk arrays. However, only one node at a time may access the data for a given group of disks. In the figure, node 1 is shown with exclusive access to the top two disks (solid line), and node 2 is shown as connected without access to the top disks (dotted line). Similarly, node 2 is shown with exclusive access to the bottom two disks (solid line), and node 1 is shown as connected without access to the bottom disks (dotted line).

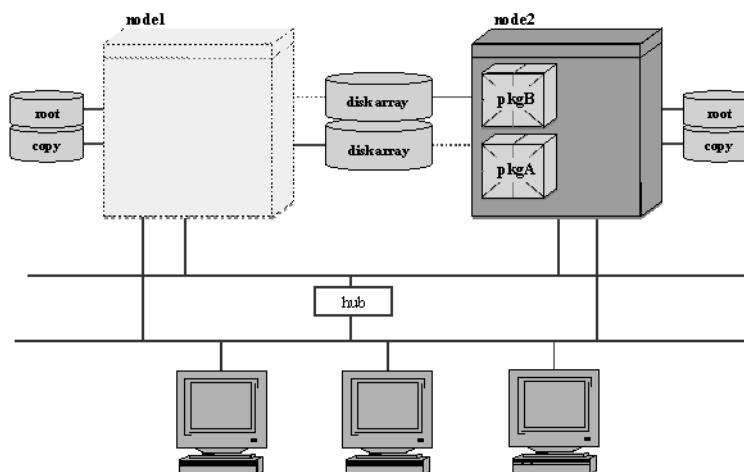
Disk arrays provide redundancy in case of disk failures. In addition, a total of four data buses are shown for the disks that are connected to node 1 and node 2. This configuration provides the maximum redundancy and also gives optimal I/O performance, since each package is using different buses.

Note that the network hardware is cabled to provide redundant LAN interfaces on each node. Serviceguard uses TCP/IP network services for reliable communication among nodes in the cluster, including the transmission of **heartbeat messages**, signals from each functioning node which are central to the operation of the cluster. TCP/IP services also are used for other types of inter-node communication. (The heartbeat is explained in more detail in the chapter “Understanding Serviceguard Software.”)

## Failover

Under normal conditions, a fully operating Serviceguard cluster simply monitors the health of the cluster's components while the packages are running on individual nodes. Any host system running in the Serviceguard cluster is called an **active node**. When you create the package, you specify a **primary node** and one or more **adoptive nodes**. When a node or its network communications fails, Serviceguard can transfer control of the package to the next available adoptive node. This situation is shown in Figure 1-2.

**Figure 1-2** Typical Cluster After Failover



After this transfer, the package typically remains on the adoptive node as long as the adoptive node continues running. If you wish, however, you can configure the package to return to its primary node as soon as the primary node comes back online. Alternatively, you may manually transfer control of the package back to the primary node at the appropriate time.

Figure 1-2 does not show the power connections to the cluster, but these are important as well. In order to remove all single points of failure from the cluster, you should provide as many separate power circuits as needed to prevent a single point of failure of your nodes, disks and disk mirrors. Each power circuit should be protected by an uninterruptible power source. For more details, refer to the section on “Power Supply Planning” in Chapter 4, “Planning and Documenting an HA Cluster.”

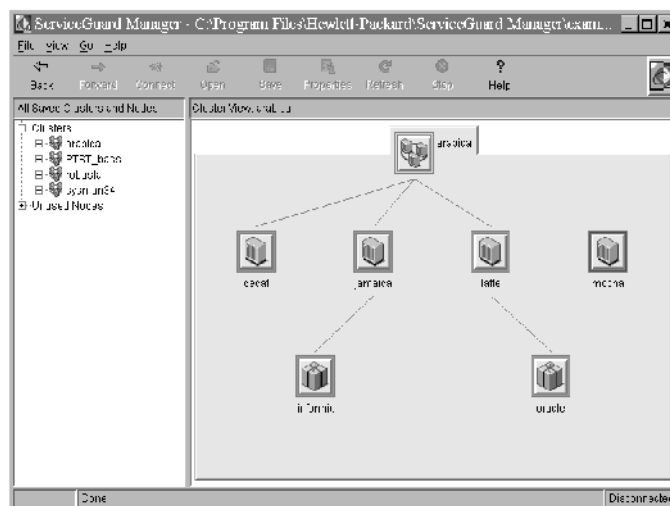
Serviceguard is designed to work in conjunction with other high availability products, such as disk arrays, which use various RAID levels for data protection; and HP-supported uninterruptible power supplies (UPS), such as HP PowerTrust, which eliminates failures related to power outage. These products are highly recommended along with Serviceguard to provide the greatest degree of availability.

---

## Viewing Serviceguard Clusters

Serviceguard Manager is a graphical user interface for creating, updating or viewing the configuration of Serviceguard clusters and observing the status of objects such as clusters, nodes and packages. Use Serviceguard Manager to take and save snapshots of cluster status to document your configurations and to provide a basis for comparison at a later date. Figure 1-3 shows a sample display of a cluster map and hierarchical view of the objects in one cluster.

**Figure 1-3** Serviceguard Manager Cluster Status



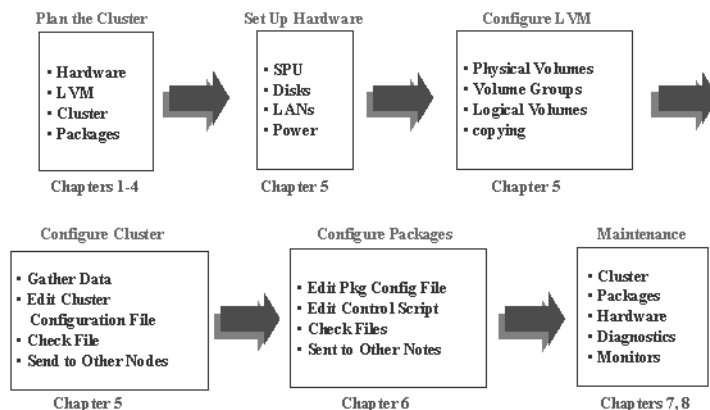
To run Serviceguard Manager, use the `sgmgr` command in Linux, or click on the Serviceguard Manager icon in Windows. For more information about Serviceguard Manager, see “Using Serviceguard Manager” in Chapter 7.

---

## Configuration Roadmap

This manual presents the tasks you need to perform in order to create a functioning HA cluster using Serviceguard. These tasks are shown in Figure 1-4.

**Figure 1-4**      **Tasks in Configuring a Serviceguard Cluster**



The tasks in Figure 1-4 are covered in step-by-step detail in chapters 4 through 7. It is strongly recommended that you gather all the data that is needed for configuration *before you start*. Refer to Chapter 4, “Planning and Documenting an HA Cluster,” for tips on gathering data.





## 2

# Understanding Hardware Configurations for Serviceguard for Linux

This chapter gives a broad overview of how the server hardware components operate with Serviceguard for Linux. The following topics are presented:

- Redundancy of Cluster Components
- Redundant Network Components
- Redundant Disk Storage
- Redundant Power Supplies

Refer to the next chapter for information about Serviceguard software components.

## Redundancy of Cluster Components

In order to provide a high level of availability, a typical cluster uses redundant system components, for example two or more SPUs and two or more independent disks. Redundancy eliminates single points of failure. In general, the more redundancy, the greater your access to applications, data, and supportive services in the event of a failure. In addition to hardware redundancy, you need software support to enable and control the transfer of your applications to another SPU or network after a failure. Serviceguard provides this support as follows:

- In the case of LAN failure, the Linux bonding facility provides a standby LAN, or Serviceguard moves packages to another node.
- In the case of SPU failure, your application is transferred from a failed SPU to a functioning SPU automatically and in a minimal amount of time.
- For software failures, an application can be restarted on the same node or another node with minimum disruption.

Serviceguard also gives you the advantage of easily transferring control of your application to another SPU in order to bring the original SPU down for system administration, maintenance, or version upgrades.

The current maximum number of nodes supported in a Serviceguard Linux cluster is 16 depending on configuration. SCSI disk arrays can be connected to a maximum of four nodes at a time on a shared (multi-initiator) bus; FibreChannel connection lets you employ up to 16 nodes. HP-supported disk arrays can be simultaneously connected to multiple nodes.

The guidelines for package failover depend on the type of disk technology in the cluster. For example, a package that accesses data on a SCSI disk array can fail over to a maximum of four nodes. A package that accesses data from a disk in a cluster using FibreChannel disk technology can be configured for failover among 16 nodes.

A package that does *not* access data from a disk on a shared bus can be configured to fail over to as many nodes as you have configured in the cluster, (currently a maximum of 16), regardless of disk technology. For instance, if a package only runs local executables, it can be configured to fail over to all nodes in the cluster that have local copies of those executables, regardless of the type of disk connectivity.

---

## Redundant Network Components

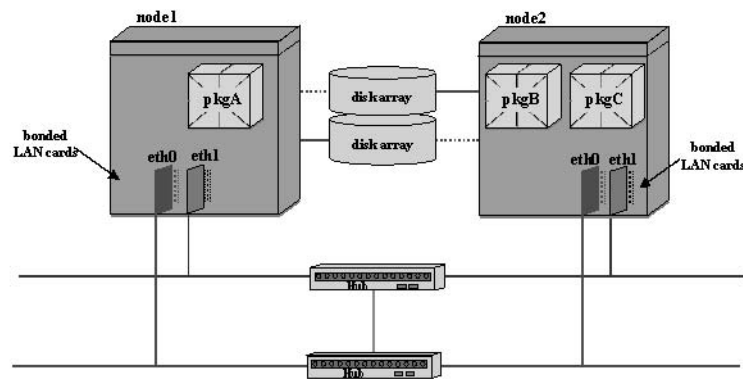
To eliminate single points of failure for networking, each subnet accessed by a cluster node is required to have redundant network interfaces. Redundant cables are also needed to protect against cable failures. Each interface card is connected to a different cable and hub or switch.

Network interfaces are allowed to share IP addresses through a process known as **channel bonding**. A sample network configuration is described further in the following sections. For a complete list of supported networks, refer to the HP Serviceguard Release Notes for your version of the product.

### Redundant Ethernet Configuration

The use of redundant network components is shown in Figure 2-1, which is an Ethernet configuration.

**Figure 2-1** Redundant LANs



In Linux configurations, the use of symmetrical LAN configurations is strongly recommended, with the use of redundant hubs or switches to connect Ethernet segments. The software bonding configurations also should be identical on both nodes, with the active interfaces being connected to the same hub or switch. (Bonding configuration is described in detail in Chapter 5.)

## Redundant Disk Storage

Each node in a cluster has its own root disk, but each node may also be physically connected to several other disks in such a way that more than one node can obtain access to the data and programs associated with a package it is configured for. This access is provided by the Logical Volume Manager (LVM). A volume group must be activated by no more than one node at a time, but when the package is moved, the volume group can be activated by the adoptive node.

---

**NOTE**

As of release A.11.16.07, Serviceguard for Linux provides functionality similar to HP-UX exclusive activation. This feature is based on LVM2 hosttags, and is available only for Linux distributions that officially support LVM2.

---

All of the disks in the volume group owned by a package must be connected to the original node and to all possible adoptive nodes for that package.

Shared disk storage in Serviceguard Linux clusters is provided by disk arrays, which have redundant power and the capability for connections to multiple nodes. Disk arrays use RAID modes to provide redundancy.

## Supported Disk Interfaces

The following interfaces are supported by Serviceguard for disks that are connected to two or more nodes (shared data disks):

- MSA500 (Modular Smart Array) family Storage
- FibreChannel.

Use the multipath mechanism that is defined in the documentation or the configuration guide of the specific storage and HBA that is used.

## Disk Monitoring

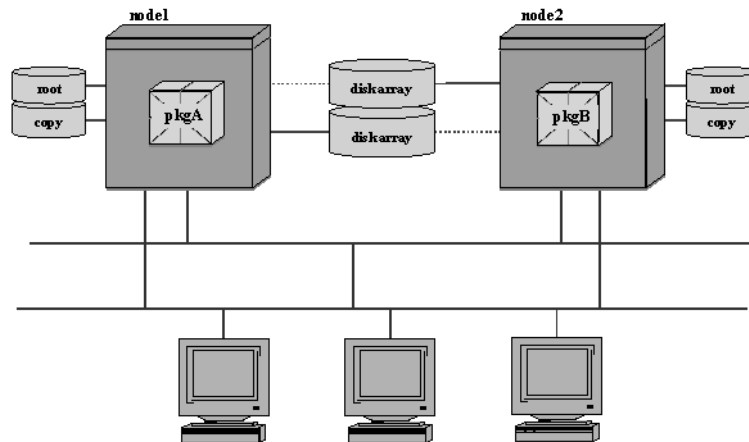
You can configure monitoring for disks and configure packages to be dependent on the monitor. For each package, you define a package service that monitors the disks that are activated by that package. If a

disk failure occurs on one node, the monitor will cause the package to fail, with the potential to fail over to a different node on which the same disks are available.

## Sample Disk Configurations

Figure 2-2 shows a two node cluster. Each node has one root disk which is mirrored and one package for which it is the primary node. Resources have been allocated to each node so that each node may adopt the package from the other node. Each package has one disk volume group assigned to it and the logical volumes in that volume group are mirrored.

**Figure 2-2**      **Mirrored Disks Connected for High Availability**



---

## **Redundant Power Supplies**

You can extend the availability of your hardware by providing battery backup to your nodes and disks. HP-supported uninterruptible power supplies (UPS), such as HP PowerTrust, can provide this protection from momentary power loss.

Disks should be attached to power circuits in such a way that disk array copies are attached to different power sources. The boot disk should be powered from the same circuit as its corresponding node. Quorum server systems should be powered separately from cluster nodes. Your HP representative can provide more details about the layout of power supplies, disks, and LAN hardware for clusters.

# 3 Understanding Serviceguard Software Components

This chapter gives a broad overview of how the Serviceguard software components work. The following topics are presented:

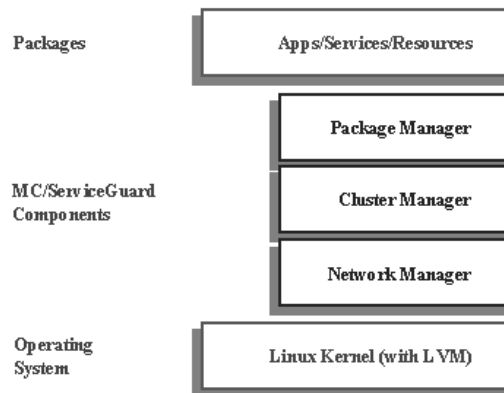
- Serviceguard Architecture
- How the Cluster Manager Works
- How the Package Manager Works
- How Package Control Scripts Work
- How the Network Manager Works
- Volume Managers for Data Storage
- Responses to Failures

If you are ready to start setting up Serviceguard clusters, skip ahead to Chapter 4, “Planning and Documenting an HA Cluster.”

## Serviceguard Architecture

The following figure shows the main software components used by Serviceguard for Linux. This chapter discusses these components in some detail.

**Figure 3-1**      **Serviceguard Software Components on Linux**



## Serviceguard Daemons

The following daemon processes are associated with Serviceguard for Linux:

- `cmclconfd`—Configuration Daemon
- `cmclld`—Cluster Daemon
- `cmlogd`—Cluster System Log Daemon
- `cmlockdiskd`—Cluster Lock LUN Daemon
- `cmomd`—Cluster Object Manager Daemon
- `cmsrvassistd`—Service Assistant Daemon
- `cmresmond`—Resource Monitor Daemon
- `qs`—Quorum Server Daemon



Each of these daemons logs to the Linux system logging files. The quorum server daemon logs to the user specified log file, such as, `/usr/local/qs/log/qs.log` file on Red Hat or `/var/log/qs/sq.log` on SuSE and `cmomd` logs to `/usr/local/cmom/log/cmomd.log` on Red Hat or `/var/log/cmom/log/cmomd.log` on SuSE.

### Configuration Daemon: `cmclconfd`

This daemon is used by the Serviceguard commands to gather information from all the nodes within the cluster. It gathers configuration information such as information on networks and volume groups. It also distributes the cluster binary configuration file to all nodes in the cluster. This daemon is started by the internet daemon, `xinetd(1M)`.

The path for this daemon is: `$SGLBIN/cmclconfd`. Relevant parameters are found in the `/etc/xinetd.d/hacl-cfg` and `/etc/xinetd.d/hacl-cfgudp` files.

---

#### NOTE

The file `/etc/cmcluster.conf` contains the mappings that resolve symbolic references to `$SGCONF`, `$SGROOT`, `$SGLBIN`, etc. See “Understanding the Location of Serviceguard Files” on page 122 for details.

---

### Cluster Daemon: `cmclsd`

This daemon is used to determine cluster membership by sending heartbeat messages to other `cmclsd` daemons on other nodes within the Serviceguard cluster. It runs at a real time priority and is locked in memory. The `cmclsd` daemon sets a **safety timer** in the kernel which is used to detect kernel hangs. If this timer is not reset periodically by `cmclsd`, the kernel will cause a system TOC, that is, a Transfer of Control, which means a CPU reset. This could occur because `cmclsd` could not communicate with the majority of the cluster’s members, or because `cmclsd` exited unexpectedly, aborted, or was unable to run for a significant amount of time and was unable to update the kernel timer, indicating a kernel hang. Before a TOC due to the expiration of the safety timer, messages will be written to the `syslog` file and the kernel’s message buffer.

The `cmclld` daemon also detects the health of the networks on the system. Finally, this daemon handles the management of Serviceguard packages, determining where to run them and when to start them. The path for this daemon is: `$SGLBIN/cmclld`.

---

**NOTE**

The file `/etc/cmcluster.conf` contains the mappings that resolve the symbolic references to `$SGCONF`, `$SGROOT`, `$SGLBIN` etc., used in this manual. See “Understanding the Location of Serviceguard Files” on page 122 for details.

---

**NOTE**

The three central components of Serviceguard—Package Manager, Cluster Manager, and Network Manager—run as parts of the `cmclld` daemon. This daemon runs as a 94 priority process and is in the `SCHED_RR` class. No other process is allowed to be a higher priority real-time process.

---

**Log Daemon: `cmlogd`**

`cmlogd` is used by `cmclld` to write messages to the system log file. Any message written to the system log by `cmclld` is written through `cmlogd`. This is to prevent any delays in writing to `syslog` from impacting the timing of `cmclld`. The path for this daemon is: `$SGLBIN/cmlogd`.

**Lock LUN Daemon: `cmlockdiskd`**

`cmlockdiskd` provides tie-breaking services when needed during cluster re-formation. The lock LUN daemon runs on each node in the cluster and is started by `cmclld` when the node joins the cluster. The lock LUN daemon on each node uses the same physical LUN in order to determine which node will continue in a tie-breaking situation.

Each member of the cluster will initiate and maintain a connection to the `cmlocklistd` daemon locally. If the `cmlockdiskd` daemon dies it will be restarted by `cmclld` and the node will reconnect to it. If there is a cluster reconfiguration while `cmlockdiskd` is down and there is a partition in the cluster that requires tie-breaking, the reconfiguration will fail. The path for this daemon is: `$SGLBIN/cmlockdiskd`.

### **Cluster Object Manager Daemon: cmomd**

This daemon is responsible for providing information about the cluster to clients—external products or tools such as Serviceguard Manager that depend on knowledge of the state of cluster objects. Clients send queries to `cmomd` and receive responses from it. This daemon may not be running on your system; it is used only by clients of the object manager.

`cmomd` accepts connections from clients, and examines queries. The queries are decomposed into categories (of classes) which are serviced by various providers. The providers gather data from various sources, including, commonly, the `cmclconfd` daemons on all connected nodes, returning data to a central assimilation point where it is filtered to meet the needs of the exact client query. This daemon is started by `xinetd`. Relevant parameters are found in the `/etc/xinetd.d/hac1-probe` file. The path for this daemon is: `$SGLBIN/cmomd`.

### **Service Assistant Daemon: cmsrvassistd**

This daemon forks and execs any script or processes as required by the cluster daemon, `cmclld`. There are two type of forks that this daemon carries out:

- Executing package run and halt scripts
- Launching services

For services, `cmclld` monitors the service process and, depending on the number of service retries, `cmclld` either restarts the service through `cmsrvassistd` or it causes the package to halt and moves the package to an available alternate node. The path for this daemon is: `$SGLBIN/cmsrvassistd`.

### **Resource Monitor Daemon: cmresmond**

This daemon communicates with `cmresserviced` processes that are launched by Serviceguard packages. The daemon reports on the status of the resources (e.g., disks) for which a monitoring request is received. If a device is found to be down, the `cmresserviced` process will exit, allowing the package to fail over.

Resources are configured for monitoring by examining all the package control scripts in the cluster and identifying the devices for which monitoring is desired. These are included in a configuration file (by default, this is named `$SGCONF/cmresmond_config.xml`), which the

## **Serviceguard Architecture**

monitor daemon reads when it starts up. This daemon is automatically started at each reboot or manually with the command `cmresmond --start`. The path for this daemon is: `$SGLBIN/cmresmond`.

The process for configuring disk monitoring is described in “Creating a Disk Monitor Configuration” on page 206.

### **Quorum Server Daemon: qs**

The quorum server daemon provides tie-breaking services when needed during cluster re-formation. The quorum server runs on a system external to the cluster, and it is started by the user, not by Serviceguard. It is normally started out of `/etc/inittab`, which means that it automatically re-spawns if it fails or is killed. Additionally, the quorum server can be run in a package in another cluster.

All members of the cluster initiate and maintain a connection to the quorum server. If the quorum server dies, the Serviceguard nodes will detect this and then periodically try to reconnect to the quorum server until it comes back up. If there is a cluster reconfiguration while the quorum server is down and there is a partition in the cluster that requires tie-breaking, the reconfiguration will fail. The path for this daemon is:

- For SUSE: `/opt/qs/bin/qs`
- For Red Hat: `/usr/local/qs/bin/qs`

## How the Cluster Manager Works

The **cluster manager** is used to initialize a cluster, to monitor the health of the cluster, to recognize node failure if it should occur, and to regulate the re-formation of the cluster when a node joins or leaves the cluster. The cluster manager operates as a daemon process that runs on each node. During cluster startup and re-formation activities, one node is selected to act as the **cluster coordinator**. Although all nodes perform some cluster management functions, the cluster coordinator is the central point for inter-node communication.

## Configuration of the Cluster

The system administrator sets up cluster configuration parameters and does an initial cluster startup; thereafter, the cluster regulates itself without manual intervention in normal operation. Configuration parameters for the cluster include the cluster name and nodes, networking parameters for the cluster heartbeat, cluster lock information, and timing parameters (discussed in detail in the “Planning” chapter). Cluster parameters are entered by editing the **cluster ASCII configuration file** (details are given in Chapter 5). The parameters you enter are used to build a binary configuration file which is propagated to all nodes in the cluster. This binary cluster configuration file must be the same on all the nodes in the cluster.

## Heartbeat Messages

Central to the operation of the cluster manager is the sending and receiving of **heartbeat messages** among the nodes in the cluster. Each node in the cluster exchanges heartbeat messages with the cluster coordinator over each TCP/IP network configured as a heartbeat device.

If a cluster node does not receive heartbeat messages from all other cluster nodes within the prescribed time, a cluster re-formation is initiated. At the end of the re-formation, if a new set of nodes form a cluster, that information is passed to the **package coordinator** (described further below, under “How the Package Manager Works”). Packages which were running on nodes that are no longer in the new cluster are transferred to their adoptive nodes. Note that if there is a transitory loss of heartbeat, the cluster may re-form with the same nodes

as before. In such cases, packages do not halt or switch, though the application may experience a slight performance impact during the re-formation.

If heartbeat and data are sent over the same LAN subnet, data congestion may cause Serviceguard to miss heartbeats during the period of the heartbeat timeout and initiate a cluster re-formation that would not be needed if the congestion had not occurred. To prevent this situation, it is recommended that you have a dedicated heartbeat as well as configuring heartbeat over the data network. A dedicated LAN is not required, but you may wish to use one if analysis of your networks shows a potential for loss of heartbeats in the cluster.

Multiple heartbeats are sent in parallel. It is recommended that you configure all subnets that interconnect cluster nodes as heartbeat networks, since this increases protection against multiple faults at no additional cost. Heartbeat networks are specified with the `HEARTBEAT_IP` cluster configuration parameter.

Each node sends its heartbeat message at a rate specified by the cluster heartbeat interval. The cluster heartbeat interval is set in the **cluster configuration file**, which you create as a part of cluster configuration, described fully in the chapter “Building an HA Cluster Configuration.”

## Manual Startup of Entire Cluster

A manual startup forms a cluster out of all the nodes in the cluster configuration. Manual startup is normally done the first time you bring up the cluster, after cluster-wide maintenance or upgrade, or after reconfiguration.

Before startup, the same binary cluster configuration file must exist on all nodes in the cluster. The system administrator starts the cluster with the `cmruncl` command issued from one node. The `cmruncl` command can only be used when the cluster is not running, that is, when none of the nodes is running the `cmclsd` daemon.

During startup, the cluster manager software checks to see if all nodes specified in the startup command are valid members of the cluster, are up and running, are attempting to form a cluster, and can communicate with each other. If they can, then the cluster manager forms the cluster.

## **Automatic Cluster Startup**

An automatic cluster startup occurs any time a node reboots and joins the cluster. This can follow the reboot of an individual node, or it may be when all nodes in a cluster have failed, as when there has been an extended power failure and all SPUs went down.

Automatic cluster startup will take place if the flag `AUTOSTART_CMCLD` is set to 1 in the `$SGCONF/cmcluster.rc` file. When any node reboots with this parameter set to 1, it will rejoin an existing cluster, or if none exists it will attempt to form a new cluster.

## **Dynamic Cluster Re-formation**

A dynamic re-formation is a temporary change in cluster membership that takes place as nodes join or leave a running cluster. Re-formation differs from reconfiguration, which is a permanent modification of the configuration files. Re-formation of the cluster occurs under the following conditions (not a complete list):

- An SPU or network failure was detected on an active node.
- An inactive node wants to join the cluster. The cluster manager daemon has been started on that node.
- A node has been added to or deleted from the cluster configuration.
- The system administrator halted a node.
- A node halts because of a package failure.
- A node halts because of a service failure.
- Heavy network traffic prohibited the heartbeat signal from being received by the cluster.
- The heartbeat network failed, and another network is not configured to carry heartbeat.

Typically, re-formation results in a cluster with a different composition. The new cluster may contain fewer or more nodes than in the previous incarnation of the cluster.



## Cluster Quorum to Prevent Split-Brain Syndrome

In general, the algorithm for cluster re-formation requires a **cluster quorum** of a strict majority (that is, more than 50%) of the nodes previously running. If both halves (exactly 50%) of a previously running cluster were allowed to re-form, there would be a **split-brain** situation in which two instances of the same cluster were running. In a split-brain scenario, different incarnations of an application could end up simultaneously accessing the same disks. One incarnation might well be initiating recovery activity while the other is modifying the state of the disks. Serviceguard's quorum requirement is designed to prevent a split-brain situation.

## Cluster Lock

Although a cluster quorum of more than 50% is generally required, exactly 50% of the previously running nodes may re-form as a new cluster *provided that the other 50% of the previously running nodes do not also re-form*. This is guaranteed by the use of a tie-breaker to choose between the two equal-sized node groups, allowing one group to form the cluster and forcing the other group to shut down. This tie-breaker is known as a **cluster lock**. The cluster lock is implemented either by means of a **lock LUN** or a **quorum server**. A cluster lock is required on two-node clusters.

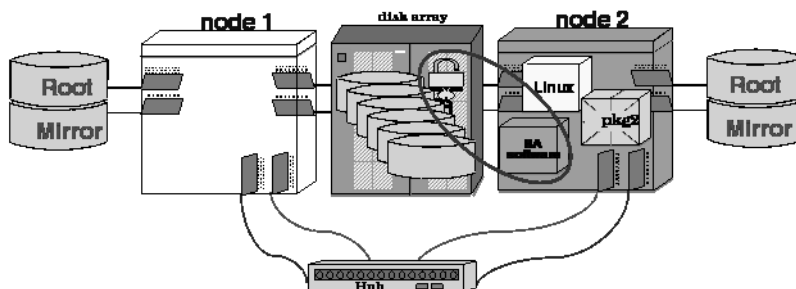
## Use of a Lock LUN as the Cluster Lock

A lock LUN may be used for clusters up to and including 4 nodes in size. The cluster lock LUN is a special area on a disk that is shareable by all nodes in the cluster. When a node obtains the cluster lock, this area is marked so that other nodes will recognize the lock as "taken."

The lock LUN is dedicated for use as the cluster lock; the disk cannot be employed as part of a normal volume with user data on it. The complete path name of the lock LUN is identified in the cluster configuration file.

The operation of the lock LUN is shown in Figure 3-2.

**Figure 3-2**      **Lock LUN Operation**



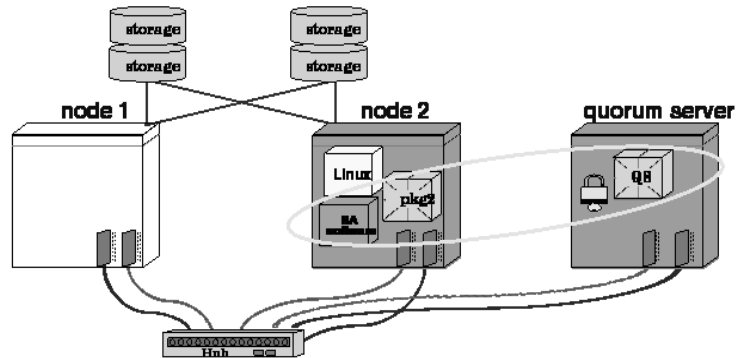
Serviceguard periodically checks the health of the lock LUN and writes messages to the syslog file when a lock disk fails the health check. This file should be monitored for early detection of lock disk problems. You can also configure the Disk Monitor to check the lock LUN.

## Use of the Quorum Server as a Cluster Lock

The cluster lock in Linux can also be implemented by means of a **quorum server**. The quorum server process runs on a machine *outside of the cluster for which it is providing quorum services*. The quorum server listens to connection requests from the Serviceguard nodes on a known port. The server maintains a special area in memory for each cluster, and when a node obtains the cluster lock, this area is marked so that other nodes will recognize the lock as “taken.” A quorum server can be used in clusters of any size. In a two-node cluster, you are required to configure a quorum server or lock LUN. If communications are lost between these two nodes, the node that obtains the cluster lock will take over the cluster and the other node will perform a TOC (transfer of control). Without a cluster lock, a failure of either node in the cluster will cause the other node, and therefore the cluster, to halt. If the quorum server is not available during an attempt to access it, the cluster will halt.

The operation of the quorum server is shown in Figure 3-3. When there is a loss of communication between node 1 and node 2, the quorum server chooses one node (in this example, node 2) to continue running in the cluster. The other node halts.

**Figure 3-3**      **Quorum Server Operation**



## Types of Quorum Server Configuration

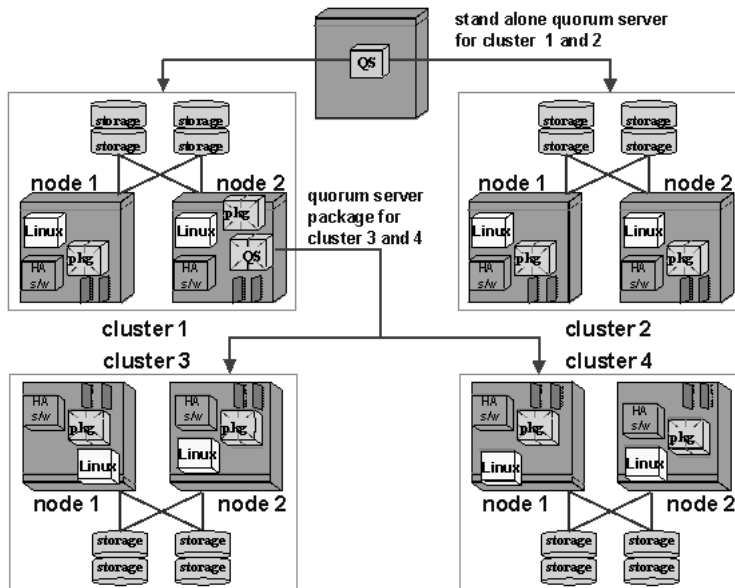
The quorum server can be configured as a Serviceguard package or as a stand alone installation. Whether it is configured in a stand alone fashion or as a Serviceguard package, the quorum server must run on a system that is separate from the cluster for which it is providing quorum services. A single quorum server, running as a package or stand alone, can provide quorum services for multiple clusters.

## How the Cluster Manager Works

If the first cluster created, in a group of clusters, needs a quorum device, that cluster must use a stand alone quorum server or lock LUN.

Figure 3-4 illustrates quorum server use across four clusters.

**Figure 3-4 Quorum Server to Cluster Distribution**



HP recommends that two clusters running quorum server packages do not provide quorum services for each other. For example if `qspkgA` running on `cluster1` is providing quorum services for `cluster2`, `qspkgB` running on `cluster2` should not provide quorum services for `cluster1`. This recommendation is a guideline, not an absolute rule.

---

## How the Package Manager Works

Each node in the cluster runs an instance of the package manager; the package manager residing on the cluster coordinator is known as the **package coordinator**.

The package coordinator does the following:

- Decides when and where to run, halt or move packages.

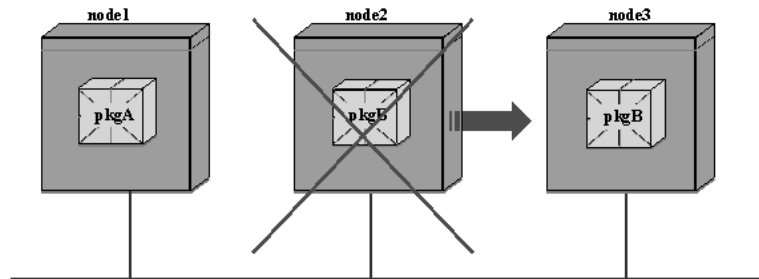
The package manager on all nodes does the following:

- Executes the user-defined control script to run and halt packages and package services.
- Reacts to changes in the status of monitored resources.

A package starts up on an appropriate node when the cluster starts. A package **failover** takes place when the package coordinator initiates the start of a package on a new node. A package failover involves both halting the existing package (in the case of a service, network, or resource failure), and starting the new instance of the package.

Failover is shown in the following figure:

**Figure 3-5**      **Package Moving During Failover**



## Configuring Packages

Each package is separately configured. You create a package by editing a **package ASCII configuration file** (detailed instructions are given in Chapter 6). Then you use the `cmapplyconf` command to check and apply the package to the cluster configuration database. You also create the **package control script**, which manages the execution of the package's services. Then the package is ready to run.

## Deciding When and Where to Run and Halt Packages

The package configuration file assigns a name to the package and includes a list of the nodes on which the package can run, in order of priority (i.e., the first node in the list is the highest priority node). In addition, the file contains three parameters that determine failover behavior. These are the `AUTO_RUN` parameter, the `FAILOVER_POLICY` parameter, and the `FAILBACK_POLICY` parameter.

## Package Switching

The `AUTO_RUN` parameter (known in earlier versions of Serviceguard as the `PKG_SWITCHING_ENABLED` parameter) defines the default global switching attribute for the package at cluster startup, that is, whether the package should be restarted automatically on a new node in response to a failure, and whether it should be started automatically when the cluster is started. Once the cluster is running, the package switching attribute of each package can be set with the `cmmodpkg` command.

The parameter is coded in the package ASCII configuration file:

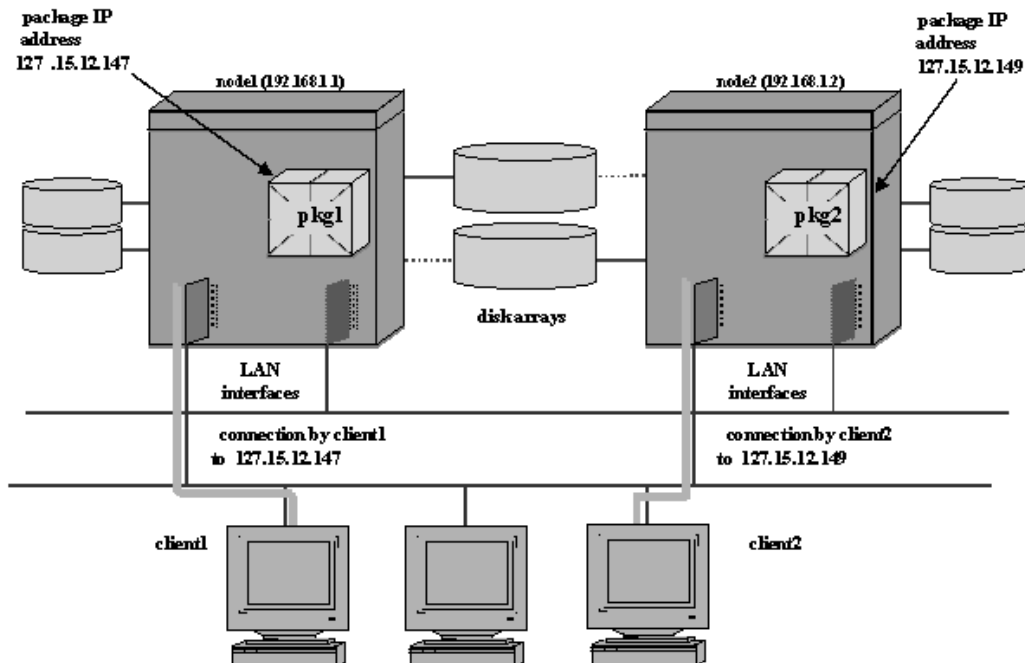
```
# The default for AUTO_RUN is YES. In the event of a
# failure, this permits the cluster software to transfer the package
# to an adoptive node. Adjust as necessary.

AUTO_RUN    YES
```

A package switch involves moving packages and their associated IP addresses to a new system. The new system must already have the same subnetwork configured and working properly, otherwise the packages will not be started. With package failovers, TCP connections are lost. TCP applications must reconnect to regain connectivity; this is not handled automatically. Note that if the package is dependent on multiple subnetworks, all subnetworks must be available on the target node before the package will be started.

The switching of relocatable IP addresses is shown in Figure 3-6 and Figure 3-7. Figure 3-6 shows a two node cluster in its original state with Package 1 running on Node 1 and Package 2 running on Node 2. Users connect to node with the IP address of the package they wish to use. Each node has a stationary IP address associated with it, and each package has an IP address associated with it.

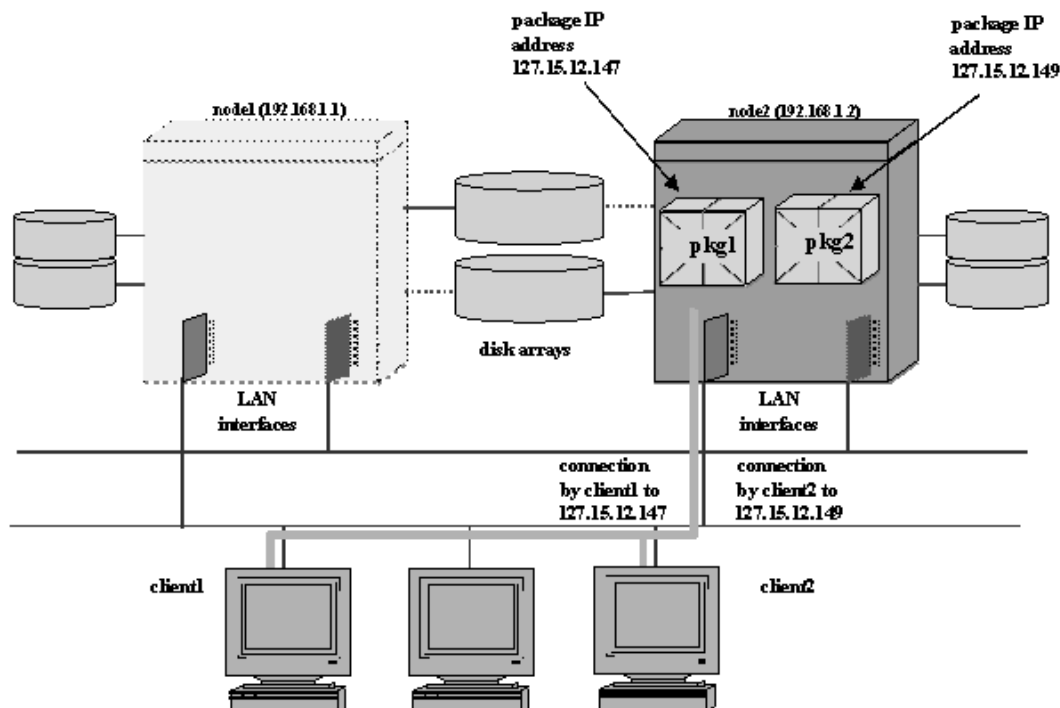
**Figure 3-6 Before Package Switching**



## How the Package Manager Works

Figure 3-7 shows the condition where Node 1 has failed and Package 1 has been transferred to Node 2. Package 1's IP address was transferred to Node 2 along with the package. Package 1 continues to be available and is now running on Node 2. Also note that Node 2 can now access both Package 1's disk and Package 2's disk.

**Figure 3-7** After Package Switching





## Failover Policy

The Package Manager selects a node for a package to run on based on the priority list included in the package configuration file together with the `FAILOVER_POLICY` parameter, also coded in the file. Failover policy governs how the package manager selects which node to run a package on when a specific node has not been identified and the package needs to be started. This applies not only to failovers but also to startup for the package, including the initial startup. The two failover policies are `CONFIGURED_NODE` (the default) and `MIN_PACKAGE_NODE`. The parameter is coded in the package ASCII configuration file:

```
# Enter the failover policy for this package. This policy will be used
# to select an adoptive node whenever the package needs to be started.
# The default policy unless otherwise specified is CONFIGURED_NODE.
# This policy will select nodes in priority order from the list of
# NODE_NAME entries specified below.

# The alternative policy is MIN_PACKAGE_NODE. This policy will select
# the node, from the list of NODE_NAME entries below, which is
# running the least number of packages at the time of failover.

#FAILOVER_POLICY          CONFIGURED_NODE
```

If you use `CONFIGURED_NODE` as the value for the failover policy, the package will start up on the highest priority node available in the node list. When a failover occurs, the package will move to the next highest priority node in the list that is available.

If you use `MIN_PACKAGE_NODE` as the value for the failover policy, the package will start up on the node that is currently running the fewest other packages. (Note that this does not mean the lightest load; the only thing that is checked is the number of packages currently running on the node.)

### Automatic Rotating Standby

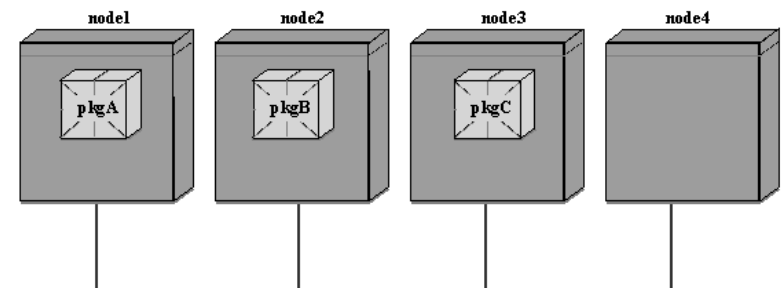
Using the MIN\_PACKAGE\_NODE failover policy, it is possible to configure a cluster that lets you use one node as an automatic rotating standby node for the cluster. Consider the following package configuration for a four node cluster. Note that all packages can run on all nodes and have the same NODE\_NAME lists. Although the example shows the node names in a different order for each package, this is not required.

**Table 3-1 Package Configuration Data**

Package Name	NODE_NAME List	FAILOVER_POLICY
pkgA	node1, node2, node3, node4	MIN_PACKAGE_NODE
pkgB	node2, node3, node4, node1	MIN_PACKAGE_NODE
pkgC	node3, node4, node1, node2	MIN_PACKAGE_NODE

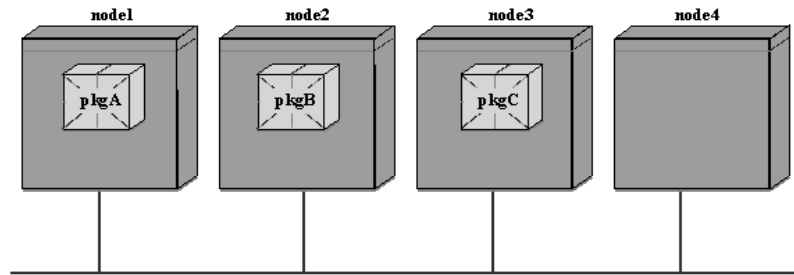
When the cluster starts, each package starts as shown in Figure 3-8.

**Figure 3-8 Rotating Standby Configuration before Failover**



If a failure occurs, any package would fail over to the node containing fewest running packages, as in Figure 3-9, which shows a failure on node 2:

**Figure 3-9 Rotating Standby Configuration after Failover**



---

**NOTE**

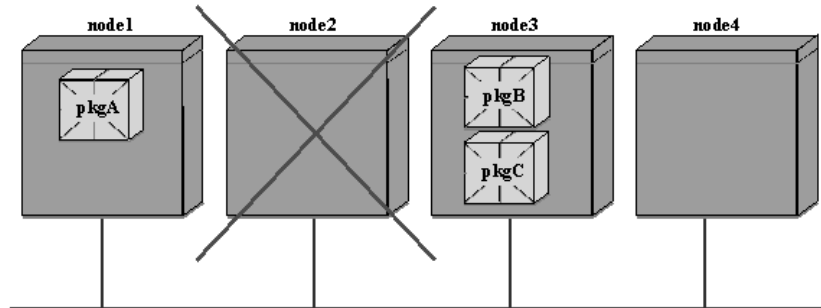
Using the `MIN_PACKAGE_NODE` policy, when node 2 is repaired and brought back into the cluster, it will then be running the fewest packages, and thus will become the new standby node.

---

If these packages had been set up using the `CONFIGURED_NODE` failover policy, they would start initially as in Figure 3-8, but the failure of node 2 would cause the package to start on node 3, as in Figure 3-10:

**Figure 3-10**

**`CONFIGURED_NODE` Policy Packages after Failover**



If you use `CONFIGURED_NODE` as the value for the failover policy, the package will start up on the highest priority node in the node list, assuming that the node is running as a member of the cluster. When a failover occurs, the package will move to the next highest priority node in the list that is available.

## Failback Policy

The use of the `FAILBACK_POLICY` parameter allows you to decide whether a package will return to its primary node if the primary node becomes available and the package is not currently running on the primary node. The configured primary node is the first node listed in the package's node list.

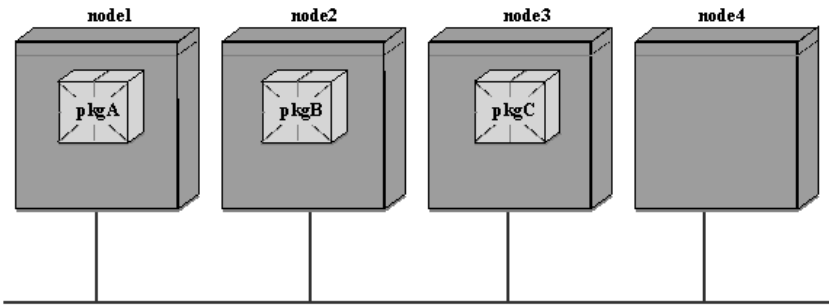
The two possible values for this policy are `AUTOMATIC` and `MANUAL`. The parameter is coded in the package ASCII configuration file:

```
# Enter the failback policy for this package. This policy will be used
# to determine what action to take during failover when a package
# is not running on its primary node and its primary node is capable
# of running the package. Default is MANUAL which means no attempt
# will be made to move the package back to its primary node when it is
# running on an alternate node. The alternate policy is AUTOMATIC which
# means the package will be moved back to its primary node whenever the
# primary node is capable of running the package.

#FAILBACK_POLICY          MANUAL
```

As an example, consider the following four-node configuration, in which `FAILOVER_POLICY` is set to `CONFIGURED_NODE` and `FAILBACK_POLICY` is `AUTOMATIC`:

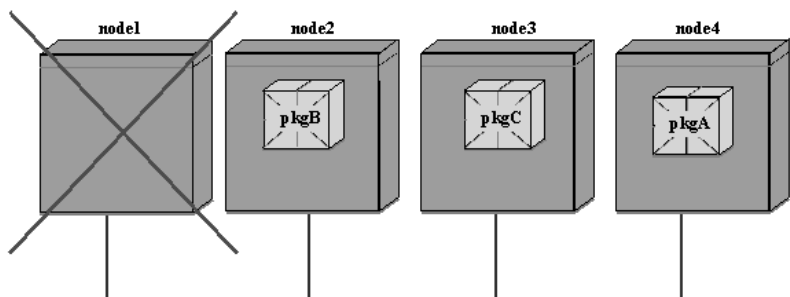
**Figure 3-11** Automatic Failback Configuration before Failover



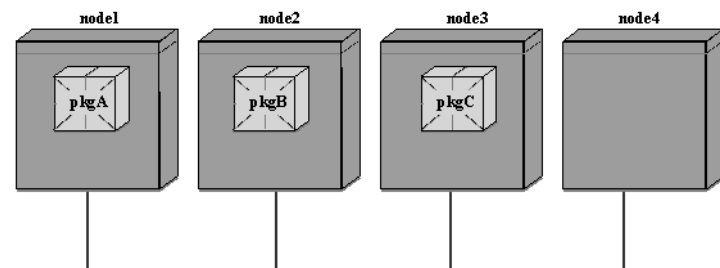
**Table 3-2** Node Lists in Sample Cluster

Package Name	NODE_NAME List	FAILOVER POLICY	FAILBACK POLICY
pkgA	node1, node4	CONFIGURED_NODE	AUTOMATIC
pkgB	node2, node4	CONFIGURED_NODE	AUTOMATIC
pkgC	node3, node4	CONFIGURED_NODE	AUTOMATIC

Node1 panics, and when the cluster reforms, pkgA starts running on node 4:

**Figure 3-12****Automatic Failback Configuration After Failover**

After rebooting, node 1 rejoins the cluster. At that point, pkgA will be automatically stopped on node 4 and restarted on node 1.

**Figure 3-13****Automatic Failback Configuration After Restart of Node 1**

---

**NOTE**

Setting the `FAILBACK_POLICY` to `AUTOMATIC` can result in a package failback and application outage during a critical production period. If you are using automatic failback, you may wish not to add the package's primary node back into the cluster until it is an appropriate time to allow the package to be taken out of service temporarily while it switches back to the primary node.

---

## On Combining Failover and Failback Policies

Combining a `FAILOVER_POLICY` of `MIN_PACKAGE_NODE` with a `FAILBACK_POLICY` of `AUTOMATIC` can result in a package's running on a node where you did not expect it to run, since the node running the fewest packages will probably not be the same host every time a failover occurs.

## Choosing Package Failover Behavior

To determine failover behavior, you can define a package failover policy that governs which nodes will automatically start up a package that is not running. In addition, you can define a failback policy that determines whether a package will be automatically returned to its primary node when that is possible.

The following table describes different types of failover behavior and the settings in the ASCII package configuration file that determine each behavior.

**Table 3-3**      **Package Failover Behavior**

Switching Behavior	Parameters in ASCII File
Package switches normally after detection of service or network failure. Halt script runs before switch takes place. (Default)	<ul style="list-style-type: none"><li>• <code>NODE_FAIL_FAST_ENABLED</code> set to NO. (Default)</li><li>• <code>SERVICE_FAIL_FAST_ENABLED</code> set to NO for all services. (Default)</li><li>• <code>AUTO_RUN</code> set to YES for the package. (Default)</li></ul>
Package fails over to the node with the fewest active packages.	<ul style="list-style-type: none"><li>• <code>FAILOVER_POLICY</code> set to <code>MIN_PACKAGE_NODE</code>.</li></ul>
Package fails over to the node that is next on the list of nodes. (Default)	<ul style="list-style-type: none"><li>• <code>FAILOVER_POLICY</code> set to <code>CONFIGURED_NODE</code>. (Default)</li></ul>
Package is automatically halted and restarted on its primary node if the primary node is available and the package is running on a non-primary node.	<ul style="list-style-type: none"><li>• <code>FAILBACK_POLICY</code> set to <code>AUTOMATIC</code>.</li></ul>

**Table 3-3                      Package Failover Behavior (Continued)**

Switching Behavior	Parameters in ASCII File
If desired, package must be manually returned to its primary node if it is running on a non-primary node.	<ul style="list-style-type: none"><li>• <code>FAILBACK_POLICY</code> set to <code>MANUAL</code>. (Default)</li><li>• <code>FAILOVER_POLICY</code> set to <code>CONFIGURED_NODE</code>. (Default)</li></ul>
All packages switch following a TOC (Transfer of Control, an immediate halt without a graceful shutdown) on the node when a specific service fails. An attempt is first made to reboot the system prior to the TOC. Halt scripts are not run.	<ul style="list-style-type: none"><li>• <code>SERVICE_FAIL_FAST_ENABLED</code> set to <code>YES</code> for a specific service.</li><li>• <code>AUTO_RUN</code> set to <code>YES</code> for all packages.</li></ul>
All packages switch following a TOC on the node when any service fails. An attempt is first made to reboot the system prior to the TOC.	<ul style="list-style-type: none"><li>• <code>SERVICE_FAIL_FAST_ENABLED</code> set to <code>YES</code> for <i>all</i> services.</li><li>• <code>AUTO_RUN</code> set to <code>YES</code> for all packages.</li></ul>



## How Package Control Scripts Work

Packages are the means by which Serviceguard starts and halts configured applications. Packages are also units of failover behavior in Serviceguard. A package is a collection of services, disk volumes and IP addresses that are managed by Serviceguard to ensure they are available. Any package can have a maximum of 30 services, and there can be as many as 150 packages per cluster. A possible total of 900 services per cluster is allowed.

### What Makes a Package Run?

A package starts up when it is not currently running, and the package manager senses that it has been enabled on an eligible node in the cluster. If there are several nodes on which the package is enabled, the package manager will use the failover policy to determine where to start the package. Note that you do not necessarily have to use a `cmrunpkg` command. In many cases, a `cmmodpkg` command that enables the package on one or more nodes is the best way to start the package.

The package manager will always try to keep the package running unless there is something preventing it from running on any node. The most common reasons for a package not being able to run are that `AUTO_RUN` is disabled, or `NODE_SWITCHING` is disabled for the package on particular nodes. When a package has failed on one node and is enabled on another node, it will start up automatically in the new location. This process is known as **package switching**, or **remote switching**.

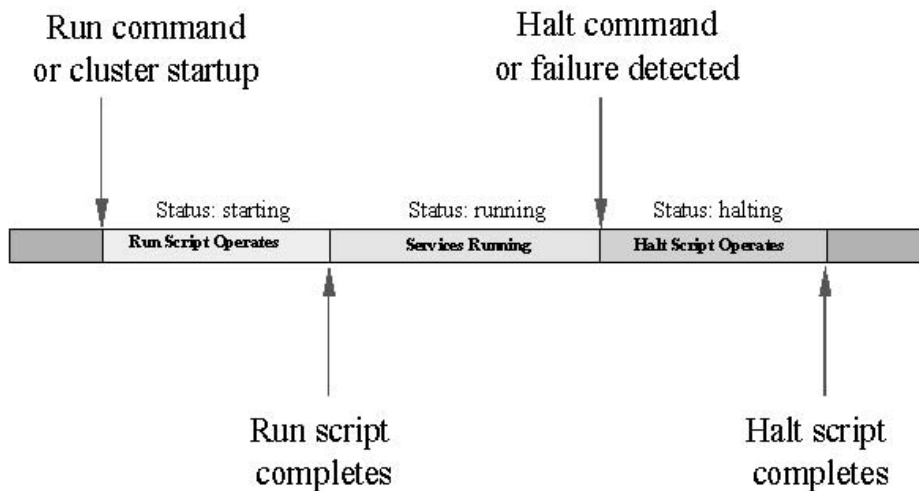
When you create the package, you indicate the list of nodes on which it is allowed to run. A standard package can run on only one node at a time, and it runs on the next available node in the node list. A package can start up automatically at cluster startup time if the `AUTO_RUN` parameter is set to `YES`. Conversely, a package with `AUTO_RUN` set to `NO` will not start automatically at cluster startup time; you must explicitly enable this kind of package using a `cmmodpkg` command.

**How Package Control Scripts Work****NOTE**

If you configure the package while the cluster is running, the package does not start up immediately after the `cmapplyconf` command completes. To start the package without halting and restarting the cluster, you must issue the `cmrunpkg` or `cmmodpkg` command.

How does the package start up, and what is its behavior while it is running? Some of the many phases of package life are shown in Figure 3-14.

**Figure 3-14 Package Time Line Showing Important Events**



The following are the most important moments in a package's life:

1. Before the control script starts
2. During run script execution
3. While services are running
4. When a service or subnet fails
5. During halt script execution
6. When the package or the node is halted with a command
7. When the node fails

## **Before the Control Script Starts**

First, a node is selected. This node must be in the package's node list, it must conform to the package's failover policy, and any resources required by the package must be available on the chosen node. One resource is the subnet that is monitored for the package. If the subnet is not available, the package cannot start on this node. Another type of resource is a dependency on a monitored external resource. If monitoring shows a value for a configured resource that is outside the permitted range, the package cannot start.

Once a node is selected, a check is then done to make sure the node allows the package to start on it. Then services are started up for a package by the control script on the selected node. Strictly speaking, the run script on the selected node is used to start the package.

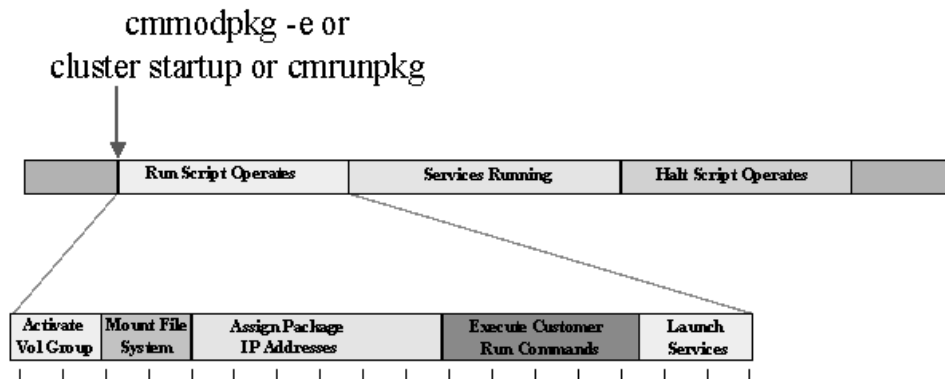
## **During Run Script Execution**

Once the package manager has determined that the package can start on a particular node, it launches the run script (that is, the control script executed with the 'start' parameter. This script carries out the following steps (also shown in Figure 3-15):

1. Activates volume groups.
2. Mounts file systems.
3. Assigns package IP addresses to the LAN card on the node.
4. Executes any customer-defined run commands.
5. Starts each package service.

6. Exits with an exit code of zero (0).

**Figure 3-15** Package Time Line for Run Script Execution



At any step along the way, an error will result in the script exiting abnormally (with an exit code of 1). For example, if a package service is unable to be started, the control script will exit with an error.

Also, if the run script execution is not complete before the time specified in the `RUN_SCRIPT_TIMEOUT`, the package manager will kill the script. During run script execution, messages are written to a log file in the same directory as the run script. This log has the same name as the run script and the extension `.log`. Normal starts are recorded in the log, together with error messages or warnings related to starting the package.

## NOTE

After the package run script has finished its work, it exits, which means that the script is no longer executing once the package is running normally. After the script exits, the PIDs of the services started by the script are monitored by the package manager directly. If the service dies, the package manager will then run the package halt script or, if `SERVICE_FAILFAST_ENABLED` is set to `YES`, it will halt the node on which the package is running. If a number of Restarts is specified for a service in the package control script, the service may be restarted if the restart count allows it, without re-running the package run script.

## Normal and Abnormal Exits from the Run Script

Exit codes on leaving the run script determine what happens to the package next. A normal exit means the package startup was successful, but all other exits mean that the start operation did not complete successfully.

- 0—normal exit. The package started normally, so all services are up on this node.
- 1—abnormal exit, also known as NO\_RESTART exit. The package did not complete all startup steps normally. Services are killed, and the package is disabled from failing over to other nodes.
- 2—alternative exit, also known as RESTART exit. There was an error, but the package is allowed to start up on another node. You might use this kind of exit from a customer defined procedure if there was an error, but starting the package on another node might succeed. A package with a RESTART exit is disabled from running on the local node, but can still run on other nodes.
- Timeout—Another type of exit occurs when the RUN\_SCRIPT\_TIMEOUT is exceeded. In this scenario, the package is killed and disabled globally. It is not disabled on the current node, however.

## Service Startup with cmrunserv

Within the package control script, the cmrunserv command starts up the individual services. This command is executed once for each service that is coded in the file. Each service has a number of restarts associated with it. The cmrunserv command passes this number to the package manager, which will restart the service the appropriate number of times if the service should fail. The following are some typical settings:

```
SERVICE_RESTART[0]=" "           ; do not restart
SERVICE_RESTART[0]="-r <n>"      ; restart as many as <n> times
SERVICE_RESTART[0]="-R"          ; restart indefinitely
```

---

**NOTE**

If you set `<n>` restarts and also set `SERVICE_FAILFAST_ENABLED` to `YES`, the failfast will take place after `<n>` restart attempts have failed. It does *not* make sense to set `SERVICE_RESTART` to `"-R"` for a service and also set `SERVICE_FAILFAST_ENABLED` to `YES`.

---

## Disk Monitor Services

Services that launch resource monitors such as the disk monitor behave just like any other service. The `SERVICE_CMD` that is used for disk monitoring is the `cmresserviced` command, as in the following example:

```
SERVICE_NAME[0]="cmresserviced_Pkg1"
SERVICE_CMD[0]="cmresserviced /dev/sdd1 /dev/sde1"
SERVICE_RESTART[0]=" "
```

The `cmresserviced` command communicates with the resource monitor daemon, `cmresmond`, which is started via `rc` scripts at each reboot or manually with the command `cmresmond --start`.

The process for configuring disk monitoring is described in “Creating a Disk Monitor Configuration” on page 206.

## While Services are Running

During the normal operation of cluster services, the package manager continuously monitors the process IDs of configured services. If a service fails but the `RESTART` parameter for that service is set to a value greater than 0, the service will restart, up to the configured number of restarts, without halting the package. During normal operation, while all services are running, you can see the status of the services in the “Script Parameters” section of the output of the `cmviewcl` command.

## When a Service or Subnet Fails

What happens when something goes wrong? If a service fails and there are no more restarts, or if the control script fails, then the package will halt on its current node and, depending on the setting of the package switching flags, may be restarted on another node. Package halting normally means that the package halt script executes (see the next section). However, if `SERVICE_FAILFAST_ENABLED` is set to yes for the service that fails, then the node will halt as soon as the failure is detected. If this flag is not set, the loss of a service will result in halting the package gracefully by running the halt script.

If `AUTO_RUN` is set to YES, the package will start up on another eligible node, if it meets all the requirements for startup. If `AUTO_RUN` is set to NO, then the package simply halts without starting up anywhere else.

## When a Package is Halted with a Command

The Serviceguard `cmhaltpkg` command has the effect of executing the package halt script, which halts the services that are running for a specific package. This provides a graceful shutdown of the package that is followed by disabling automatic package startup (`AUTO_RUN`).

---

### NOTE

If the `cmhaltpkg` command is issued with the `-n <nodename>` option, then the package is halted only if it is running on that node.

---

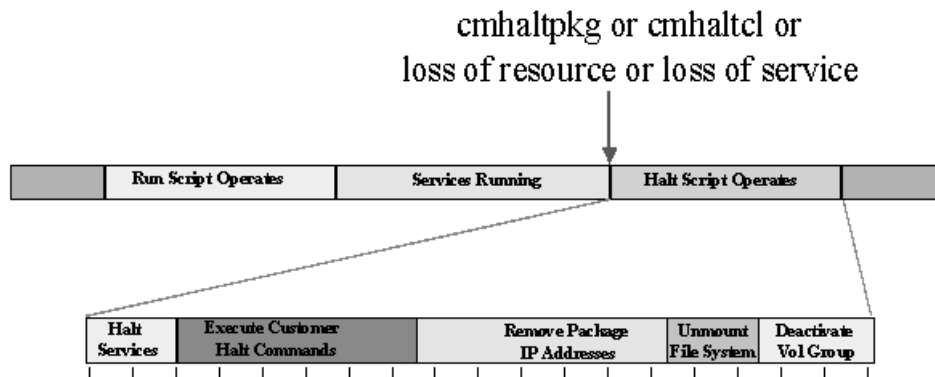
The `cmmodpkg` command cannot be used to halt a package, but it can disable switching either on particular nodes or on all nodes. A package can continue running when its switching has been disabled, but it will not be able to start on other nodes if it stops running on its current node.

## During Halt Script Execution

Once the package manager has detected a service failure, or when the `cmhaltpkg` command has been issued for a particular package, then it launches the halt script (that is, the control script executed with the 'halt' parameter. This script carries out the following steps (also shown in Figure 3-16):

1. Halts all package services.
2. Executes any customer-defined halt commands.
3. Removes package IP addresses from the LAN card on the node.
4. Unmounts file systems.
5. Deactivates volume groups.
6. Exits with an exit code of zero (0).

**Figure 3-16** Package Time Line for Halt Script Execution



At any step along the way, an error will result in the script exiting abnormally (with an exit code of 1). Also, if the halt script execution is not complete before the time specified in the `HALT_SCRIPT_TIMEOUT`, the package manager will kill the script. During halt script execution, messages are written to a log file in the same directory as the halt script. This log has the same name as the halt script and the extension `.log`. Normal starts are recorded in the log, together with error messages or warnings related to halting the package.



## Normal and Abnormal Exits from the Halt Script

The package's ability to move to other nodes is affected by the exit conditions on leaving the halt script. The following are the possible exit codes:

- 0—normal exit. The package halted normally, so all services are down on this node.
- 1—abnormal exit, also known as NO\_RESTART exit. The package did not halt normally. Services are killed, and the package is disabled globally. It is not disabled on the current node, however.
- Timeout—Another type of exit occurs when the HALT\_SCRIPT\_TIMEOUT is exceeded. In this scenario, the package is killed and disabled globally. It is not disabled on the current node, however.

### Package Control Script Error and Exit Conditions

Table 3-4 shows the possible combinations of error condition, failfast setting and package movement for failover packages.

**Table 3-4 Error Conditions and Package Movement**

Package Error Condition			Results			
Error or Exit Code	Node Failfast Enabled	Service Failfast Enabled	Linux Status on Primary after Error	Halt script runs after Error or Exit	Package Allowed to Run on Primary Node after Error	Package Allowed to Run on Alternate Node
Service Failure	YES	YES	TOC	No	N/A (TOC)	Yes
Service Failure	NO	YES	TOC	No	N/A (TOC)	Yes
Service Failure	YES	NO	Running	Yes	No	Yes
Service Failure	NO	NO	Running	Yes	No	Yes

**Table 3-4 Error Conditions and Package Movement (Continued)**

<b>Package Error Condition</b>			<b>Results</b>			
<b>Error or Exit Code</b>	<b>Node Failfast Enabled</b>	<b>Service Failfast Enabled</b>	<b>Linux Status on Primary after Error</b>	<b>Halt script runs after Error or Exit</b>	<b>Package Allowed to Run on Primary Node after Error</b>	<b>Package Allowed to Run on Alternate Node</b>
Run Script Exit 1	Either Setting	Either Setting	Running	No	Not changed	No
Run Script Exit 2	YES	Either Setting	TOC	No	N/A (TOC)	Yes
Run Script Exit 2	NO	Either Setting	Running	No	No	Yes
Run Script Timeout	YES	Either Setting	TOC	No	N/A (TOC)	Yes
Run Script Timeout	NO	Either Setting	Running	No	Not changed	No
Halt Script Exit 1	YES	Either Setting	Running	N/A	Yes	No
Halt Script Exit 1	NO	Either Setting	Running	N/A	Yes	No

**Table 3-4 Error Conditions and Package Movement (Continued)**

Package Error Condition			Results			
Error or Exit Code	Node Failfast Enabled	Service Failfast Enabled	Linux Status on Primary after Error	Halt script runs after Error or Exit	Package Allowed to Run on Primary Node after Error	Package Allowed to Run on Alternate Node
Halt Script Timeout	YES	Either Setting	TOC	N/A	N/A (TOC)	Yes, unless the timeout happened after the <code>cmhaltpkg</code> command was executed.
Halt Script Timeout	NO	Either Setting	Running	N/A	Yes	No
Service Failure	Either Setting	YES	TOC	No	N/A (TOC)	Yes
Service Failure	Either Setting	NO	Running	Yes	No	Yes
Loss of Network	YES	Either Setting	TOC	No	N/A (TOC)	Yes
Loss of Network	NO	Either Setting	Running	Yes	Yes	Yes

## How the Network Manager Works

The purpose of the network manager is to detect and recover from network card and cable failures so that network services remain highly available to clients. In practice, this means assigning IP addresses for each package to the primary LAN interface card on the node where the package is running and monitoring the health of all interfaces.

### Stationary and Relocatable IP Addresses

Each node (host system) should have an IP address for each active network interface. This address, known as a **stationary IP address**, is configured in the file

`/etc/sysconfig/network-scripts/ifcfg-<interface>` on Red Hat or `/etc/sysconfig/network/ifcfg-<mac_address>` on SUSE. The stationary IP address is *not* associated with packages, and it is not transferable to another node. Stationary IP addresses are used to transmit heartbeat messages (described earlier in the section “How the Cluster Manager Works”) and other data.

In addition to the stationary IP address, you normally assign one or more unique IP addresses to each package. The package IP address is assigned to the primary LAN interface card by the `cmmodnet` command in the package control script when the package starts up. The IP addresses associated with a package are called **relocatable IP addresses** (also known as **IP aliases**, **package IP addresses** or **floating IP addresses**) because the addresses can actually move from one cluster node to another. You can use up to 200 relocatable IP addresses in a cluster spread over as many as 150 packages.

A relocatable IP address is like a virtual host IP address that is assigned to a package. It is recommended that you configure names for each package through DNS (Domain Name System). A program then can use the package's name like a host name as the input to `gethostbyname(3)`, which will return the package's relocatable IP address.

Relocatable addresses (but not stationary addresses) can be taken over by an adoptive node if control of the package is transferred. This means that applications can access the package via its relocatable address without knowing which node the package currently resides on.

## Adding and Deleting Relocatable IP Addresses

When a package is started, a relocatable IP address can be added to a specified IP subnet. When the package is stopped, the relocatable IP address is deleted from the specified subnet. Adding and removing of relocatable IP addresses is handled through the `cmmmodnet` command in the **package control script**, which is described in detail in the chapter “Configuring Packages and their Services.”

IP addresses are configured only on each primary network interface card. Multiple IP addresses on the same network card must belong to the same IP subnet.

### Load Sharing

Serviceguard allows you to configure several services into a single package, sharing a single IP address; in that case all those services will fail over when the package does. If you want to be able to load-balance services (that is, move a specific service to a less loaded system when necessary) you can do so by putting each service in its own package and giving it a unique IP address.

## Bonding of LAN Interfaces

On the local node, several LAN interfaces can be grouped together in a process known in Linux as **channel bonding**. In the bonded group, one interface is used to transmit and receive data, while the others are available as backups. If one interface fails, another interface in the bonded group takes over. It is strongly recommended to use channel bonding in each critical IP subnet to achieve highly available network services.

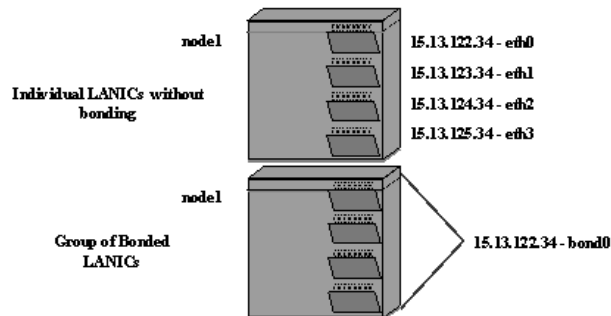
Although **Host Bus Adapters (HBAs)** do not have to be identical, bonding is supported only between Ethernet LANs of the same type. Serviceguard for Linux supports the use of bonding of LAN interfaces at the driver level. The Ethernet driver is configured to employ a group of interfaces.

Once bonding is enabled, each interface can be viewed as a single logical link of multiple physical ports with only one IP and MAC address. There is no limit to the number of slaves (ports) per bond, and the number of bonds per system is limited to the number of Linux modules you can load.

You can bond the ports within a multi-ported networking card (cards with up to four ports are currently available). Alternatively, you can bond ports from different cards. Figure 3-17 shows an example of four separate interfaces bonded into one aggregate.

**Figure 3-17**

### Bonded Network Interfaces

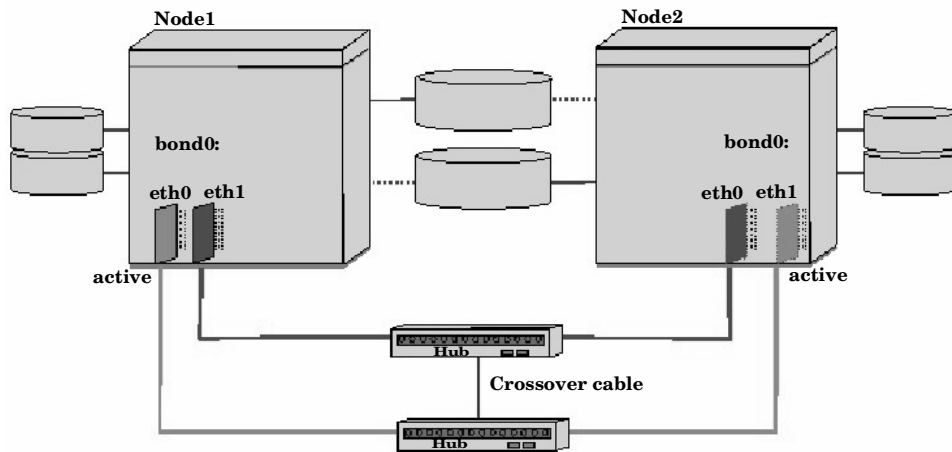


Both the Single and Dual ported LANs in the non-bonded configuration have four LAN cards, each associated with a separate non-aggregated IP address and MAC address, and each with its own LAN name (eth1, eth2, eth3, eth4). When these ports are aggregated, all four ports are

associated with a single IP address and MAC address. In this example, the aggregated ports are collectively known as `bond0`, and this is the name by which the bond is known during cluster configuration.

Figure 3-18 shows a bonded configuration using redundant hubs with a crossover cable.

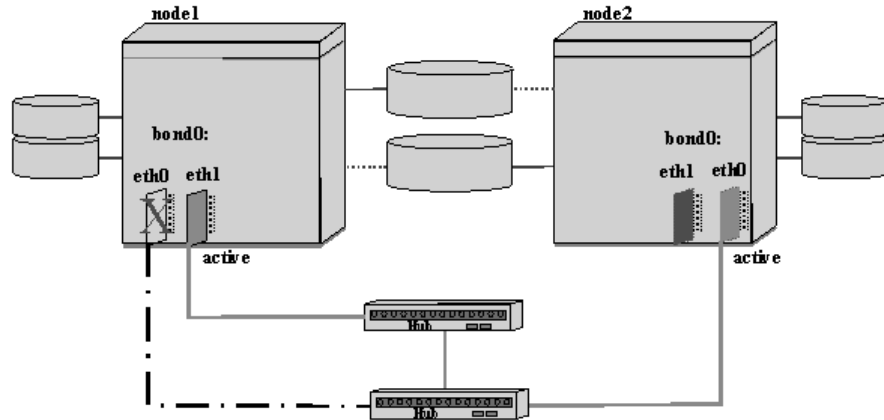
**Figure 3-18**      **Bonded NICs**



In bonding, individual Ethernet interfaces are **slaves**, and the bond is the **master**. In the basic high availability configuration (mode 1), one slave in a bond assumes an active role, while the others remain inactive until a failure is detected. (In Figure 3-18, both `eth0` slave interfaces are active.) It is important that during configuration, the active slave interfaces on all nodes are connected to the same hub. If this were not the case, then normal operation of the LAN would *require* the use of the crossover between the hubs and the crossover would become a single point of failure.

After the failure of a card, messages are still carried on the bonded LAN and are received on the other node, but now eth1 has become active in bond0 on node1. This situation is shown in Figure 3-19.

**Figure 3-19**      **Bonded NICs After Failure**



Various combinations of Ethernet card types (single or dual-port) and bond groups are possible, but it is vitally important to remember that at least two physical cards must be used in any combination of channel bonds to avoid a single point of failure for heartbeat connections.

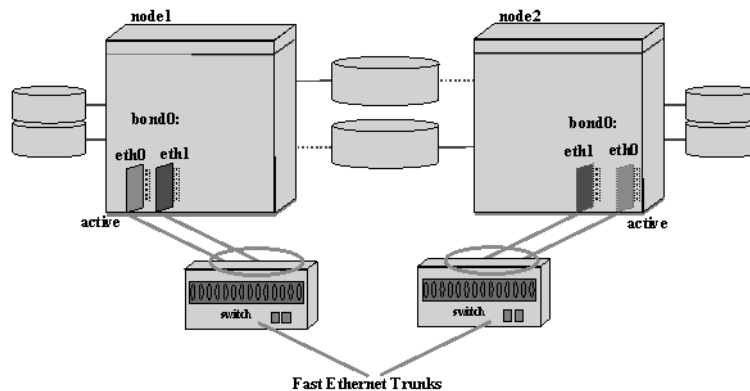


## Bonding for Load Balancing

It is also possible to configure bonds in load balancing mode, which allows all slaves to transmit data in parallel, in an active/active arrangement. In this case, high availability is provided by the fact that the bond still continues to function (with less throughput) if one of the component LANs should fail. The user should check the Bonding documentation to determine if the hardware configuration must use Ethernet switches such as the HP Procurve switch, which supports trunking of switch ports. The bonding driver configuration must specify mode 0 for the bond type.

An example of this type of configuration is shown in Figure 3-20.

**Figure 3-20**      **Bonded NICs Configured for Load Balancing**



## **Remote Switching**

A remote switch (that is, a package switch) involves moving packages and their associated IP addresses to a new system. The new system must already have the same subnetwork configured and working properly, otherwise the packages will not be started. With remote switching, TCP connections are lost. TCP applications must reconnect to regain connectivity; this is not handled automatically. Note that if the package is dependent on multiple subnetworks, all subnetworks must be available on the target node before the package will be started.

The remote switching of relocatable IP addresses was shown previously in Figure 3-6 and Figure 3-7.

## **ARP Messages after Switching**

When a floating IP address is moved to a new interface, either locally or remotely, an ARP message is broadcast to indicate the new mapping between IP address and link layer address. An ARP message is sent for each IP address that has been moved. All systems receiving the broadcast should update the associated ARP cache entry to reflect the change. Currently, the ARP messages are sent at the time the IP address is added to the new system. An ARP message is sent in the form of an ARP request. The sender and receiver protocol address fields of the ARP request message are both set to the same floating IP address. This ensures that nodes receiving the message will not send replies.

## Volume Managers for Data Storage

A volume manager is a tool that lets you create units of disk storage that are more flexible than individual disk partitions. These units can be used on single systems or in high availability clusters. HP Serviceguard for Linux uses the Linux Logical Volume Manager (LVM) which creates redundant storage groups. This section provides an overview of volume management with LVM. Refer to the section on “Creating the Logical Volume Infrastructure” in Chapter 5 for details about configuring volume groups, logical volumes, and file systems for use in Serviceguard packages.

In HP Serviceguard for Linux, the supported shared data storage type is **disk arrays** which configure redundant storage in hardware.

When you have a disk array, the basic element of storage is a LUN, which already provides storage redundancy via RAID1 or RAID5. Before you can use the LUNs, you must partition them using `fdisk`.

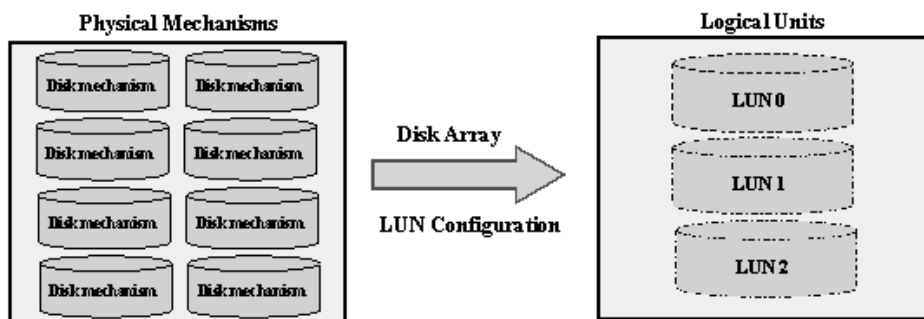
In LVM, you manipulate storage in one or more **volume groups**. A volume group is built by grouping individual **physical volumes**. Physical volumes can be disk partitions or LUNs that have been marked as physical volumes as described below.

You use the `pvccreate` command to mark the LUN as physical volumes. Then you use the `vgcreate` command to create volume groups out of one or more physical volumes. Once configured, a volume group can be subdivided into **logical volumes** of different sizes and types. File systems or databases used by the applications in the cluster are mounted on these logical volumes. In Serviceguard clusters, volume groups are **activated** by package control scripts when an application starts up, and they are **deactivated** by package control scripts when the application halts.

## Examples of Storage on Smart Arrays

Figure 3-21 shows an illustration of storage configured on a Smart MSA500 storage. Physical disks are configured by an array utility program into logical units or LUNs which are then seen by the operating system.

**Figure 3-21**      **Physical Disks Combined into LUNs**



---

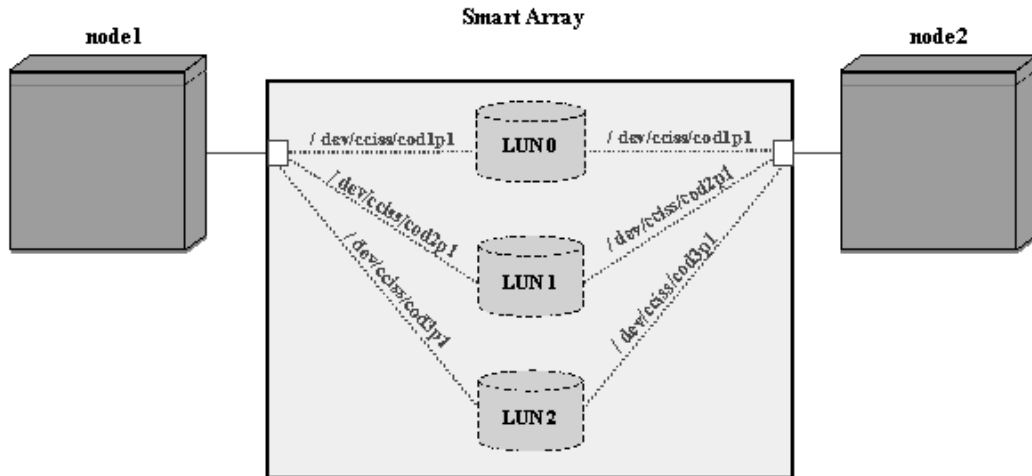
**NOTE**

LUN definition is normally done using utility programs provided by the disk array manufacturer. Since arrays vary considerably, you should refer to the documentation that accompanies your storage unit.

---

Figure 3-22 shows LUNs configured with a Smart Array cluster storage with single pathways to the data.

**Figure 3-22**      **Smart Array Single Paths to LUNs**



Finally, the Smart Array LUNs are configured into volume groups as shown in Figure 3-23.

**Figure 3-23** Smart Array LUNs Configured in Volume Groups

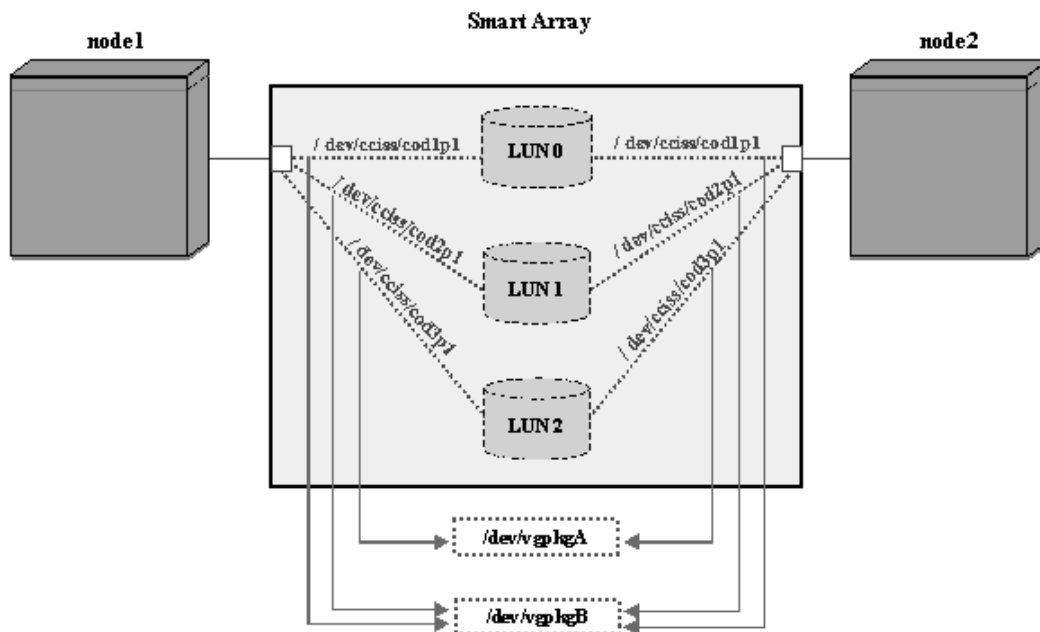
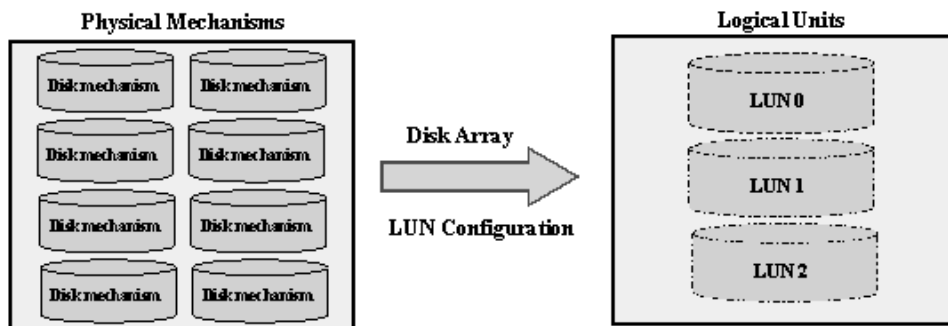


Figure 3-24 shows an illustration of storage configured on a disk array. Physical disks are configured by an array utility program into logical units or LUNs which are then seen by the operating system.

**Figure 3-24** Physical Disks Combined into LUNs



---

**NOTE**

LUN definition is normally done using utility programs provided by the disk array manufacturer. Since arrays vary considerably, make sure you read the documentation that accompanies your storage unit.

---

## Multipathing and LVM

How multipathing is implemented depends on the storage sub-system attached to the cluster and the HBA in the servers. Check the documentation that accompanied your storage sub-system and HBA.

- For fibre-channel storage, use the multipath function within the HBA driver, if that function is supported by HP.
  - For the QLogic driver, refer to the application note “Using the QLogic HBA driver for single-path or multipath failover mode on Linux systems”, which you can find by entering the terms “qlogic multipath application” in the search box of [www.hp.com](http://www.hp.com).

Once you have created the LUNs, configure each LUN into LVM as a single physical volume, following the directions in chapter 5 (the multiple paths are hidden by the driver).

---

**NOTE**

Serviceguard does not support the multiple device driver (MD) for fibre-channel storage, nor does it support the Device Mapper (DM).

---

- For MSA 500 family devices that support multipathing, use the MD driver. Use the `mdadm` command to create a multipath device, then configure the device into LVM as a single physical volume. For example, you might use the command

```
mdadm -create /dev/md0 --level=multipath --raid-devices=2 /dev/sdf1 dev/sdh1
```

to create a multipath device from devices `/dev/sdf1` and `/dev/sdh1`. Then configure `/dev/md0` into LVM as a single physical device, following the directions in chapter 5.

---

**NOTE**

Do not use the MD lines in the Serviceguard package control scripts for MD multipath. Instead, for multipath only, MD activation can be done at system boot.

---

## Monitoring Disks

The devices that are used for shared storage can be monitored by the resource monitoring facility `cmresmond`. This daemon runs on all nodes in the cluster and keeps track of the status of disks that are configured for monitoring. A monitor server allows you to view the status of disks in a standard web browser. Each package configuration includes information about the disks that are to be activated by the package at startup. If monitoring is used for a particular disk, or LUN, then the health of the disk is checked at package startup. The package will fail if the disk is not available.

The failure of a monitored disk will cause the monitor's client service to exit, with the consequent failure of the package on the node. When this happens, the package may be restarted on another node. If `AUTO_RUN` is set to `YES`, the package will start up on another eligible node, if it meets all the requirements for startup. If `AUTO_RUN` is set to `NO`, then the package simply halts without starting up anywhere else.

The process for configuring disk monitoring is described in “Creating a Disk Monitor Configuration” on page 206.

## More Information on LVM

Refer to the section “Creating the Logical Volume Infrastructure” in Chapter 5 for details about configuring volume groups, logical volumes, and file systems for use in Serviceguard packages.

Refer to the article, *Logical Volume Manager HOWTO* on the Linux Documentation Project page at <http://www.tldp.org> for a basic description of Linux LVM.



## Responses to Failures

HP Serviceguard responds to different kinds of failures in specific ways. For most hardware failures, the response is not user-configurable, but for package and service failures, you can choose the system's response, within limits.

### Transfer of Control (TOC) When a Node Fails

The most dramatic response to a failure in a Serviceguard cluster is a Linux TOC (Transfer of Control), which is an immediate halt of the SPU without a graceful shutdown. This TOC is done to protect the integrity of your data.

A TOC is done if a cluster node cannot communicate with the majority of cluster members for the pre-determined time, or if there is a kernel hang, a kernel spin, a runaway real-time process, or if the HP Serviceguard cluster daemon, `cmclld`, fails. During this event, the following message is sent to the console:

```
DEADMAN: Time expired, initiating system restart.
```

A TOC is also initiated by Serviceguard itself under specific circumstances. If the service failfast parameter is enabled in the package configuration file, the entire node will fail with a TOC whenever there is a failure of that specific service. If `NODE_FAIL_FAST_ENABLED` is set to `YES` in the package configuration file, the entire node will fail with a TOC whenever there is a timeout or a failure causing the package control script to exit with a value other than 0 or 1. In addition, a node-level failure may also be caused by events independent of a package and its services. Loss of the heartbeat or loss of the cluster daemon (`cmclld`) or other critical daemons will cause a node to fail even when its packages and their services are functioning.

In some cases, an attempt is first made to reboot the system prior to the TOC. If the reboot is able to complete before the safety timer expires, then the TOC will not take place. In either case, packages are able to move quickly to another node.

## Responses to Hardware Failures

If a serious system problem occurs, such as a system panic or physical disruption of the SPU's circuits, Serviceguard recognizes a node failure and transfers the packages currently running on that node to an adoptive node elsewhere in the cluster. The new location for each package is determined by that package's configuration file, which lists primary and alternate nodes for the package. Transfer of a package to another node does not transfer the program counter. Processes in a transferred package will restart from the beginning. In order for an application to be expeditiously restarted after a failure, it must be “crash-tolerant”; that is, all processes in the package must be written so that they can detect such a restart. This is the same application design required for restart after a normal system crash.

In the event of a LAN interface failure, bonding provides a backup path for IP messages. If a heartbeat LAN interface fails and no redundant heartbeat is configured, the node fails with a TOC. If a monitored data LAN interface fails without a standby, the node fails with a TOC only if `NODE_FAILFAST_ENABLED` (described further in the “Planning” chapter under “Package Configuration Planning”) is set to `YES` for the package.

Disk monitoring provides additional protection. You can configure packages to be dependent on the health of disks, so that when a disk monitor reports a problem, the package can fail over to another node.

Serviceguard does not respond directly to power failures, although a loss of power to an individual cluster component may appear to Serviceguard like the failure of that component, and will result in the appropriate switching behavior. Power protection is provided by HP-supported uninterruptible power supplies (UPS), such as HP PowerTrust.

## Responses to Package and Service Failures

In the default case, the failure of the package or of a service within a package causes the package to shut down by running the control script with the 'stop' parameter, and then restarting the package on an alternate node.

If you wish, you can modify this default behavior by specifying that the node should crash (TOC) before the transfer takes place. If this behavior is specified, HP Serviceguard will attempt to reboot the system prior to a TOC. If there is enough time to flush the buffers in the buffer cache, the reboot is successful, and a TOC does not take place. Either way, the system will be guaranteed to come down within a predetermined number of seconds.

In cases where package shutdown might hang, leaving the node in an unknown state, the use of a Failfast option can provide a quick failover, after which the node will be cleaned up on reboot. Remember, however, that when the node crashes, *all* packages on the node are halted abruptly.

The settings of node and service failfast parameters during package configuration will determine the exact behavior of the package and the node in the event of failure. The section on “Package Configuration Parameters” in the “Planning” chapter contains details on how to choose an appropriate failover behavior.

### Service Restarts

You can allow a service to restart locally following a failure. To do this, you indicate a number of restarts for each service in the package control script. When a service starts, the variable `RESTART_COUNT` is set in the service's environment. The service, as it executes, can examine this variable to see whether it has been restarted after a failure, and if so, it can take appropriate action such as cleanup.

### Network Communication Failure

An important element in the cluster is the health of the network itself. As it continuously monitors the cluster, each node listens for heartbeat messages from the other nodes confirming that all nodes are able to communicate with each other. If a node does not hear these messages within the configured amount of time, a node timeout occurs, resulting in a cluster re-formation and later, if there are still no heartbeat messages received, a TOC.



# 4 Planning and Documenting an HA Cluster

Building a Serviceguard cluster begins with a planning phase in which you gather and record information about all the hardware and software components of the configuration. Planning starts with a simple list of hardware and network components. As the installation and configuration continue, the list is extended and refined. After hardware installation, you can use a variety of Linux commands to obtain information about your configuration; this information is entered on the worksheets provided in this chapter. During the creation of the cluster, the planning worksheets provide the values that are edited into the configuration files and control scripts.

When the configuration is finished, you can use Serviceguard Manager to capture all the configuration data in a .sgm file, and you can display a map of the cluster, which can also be saved in a .gif file for later use. Refer to the section “Using Serviceguard Manager” in Chapter 7.

This chapter assists you in the following planning areas:

- General Planning
- Hardware Planning
- Power Supply Planning
- Quorum Server Planning
- Volume Manager Planning
- Cluster Configuration Planning
- Package Configuration Planning

The description of each planning step in this chapter is accompanied by a worksheet on which you can optionally record the parameters and other data relevant for successful setup and maintenance. As you go through each step, record all the important details of the configuration so as to document your production system. During the actual configuration of the cluster, refer to the information from these worksheets. A complete set of blank worksheets is in Appendix D.

---

**NOTE**

Planning and installation overlap considerably, so you may not be able to complete the worksheets before you proceed to the actual configuration. In cases where the worksheet is incomplete, fill in the missing elements to document the system as you proceed with the configuration.

---

Subsequent chapters describe configuration and maintenance tasks in detail.

## General Planning

A clear understanding of your high availability objectives will quickly help you to define your hardware requirements and design your system. Use the following questions as a guide for general planning:

1. What applications must continue to be available in the event of a failure?
2. What system resources (processing power, networking, SPU, memory, disk space) are needed to support these applications?
3. How will these resources be distributed among the nodes in the cluster during normal operation?
4. How will these resources be distributed among the nodes of the cluster in all possible combinations of failures, especially node failures?
5. How will resources be distributed during routine maintenance of the cluster?
6. What are the networking requirements? Are all networks and subnets available?
7. Have you eliminated all single points of failure? For example:
  - network points of failure.
  - disk points of failure.
  - electrical points of failure.
  - application points of failure.

## Serviceguard Memory Requirements

The amount of lockable memory required for Serviceguard for Linux depends on the number of packages and resources configured in the cluster. To calculate a rough estimate of how much lockable memory is needed:

Total Memory = 6 MB (for the HP Serviceguard for Linux base) + 80 KB (for each package in the cluster)

For example, if you have 10 packages in the cluster, you would require 800 KB (80x10), plus 6 MB for a total of 6.80 MB.

The total amount is needed *on each node in the cluster*, regardless of whether a given package or resource is on that node or not.

## Planning for Expansion

When you first set up the cluster, you indicate a set of nodes and define a group of packages for the initial configuration. At a later time, you may wish to add additional nodes and packages, or you may wish to use additional disk hardware for shared data storage. If you intend to expand your cluster *without the need to bring it down*, careful planning of the initial configuration is required. Use the following guidelines:

- Set the Maximum Configured Packages parameter (described later in this chapter in the “Cluster Configuration Planning” section) high enough to accommodate the additional packages you plan to add. Keep in mind that adding package capacity uses memory resources (80K per package).
- Networks should be pre-configured into the cluster configuration if they will be needed for packages you will add later while the cluster is running.

Refer to the chapter on “Cluster and Package Maintenance” for more information about changing the cluster configuration dynamically, that is, while the cluster is running.

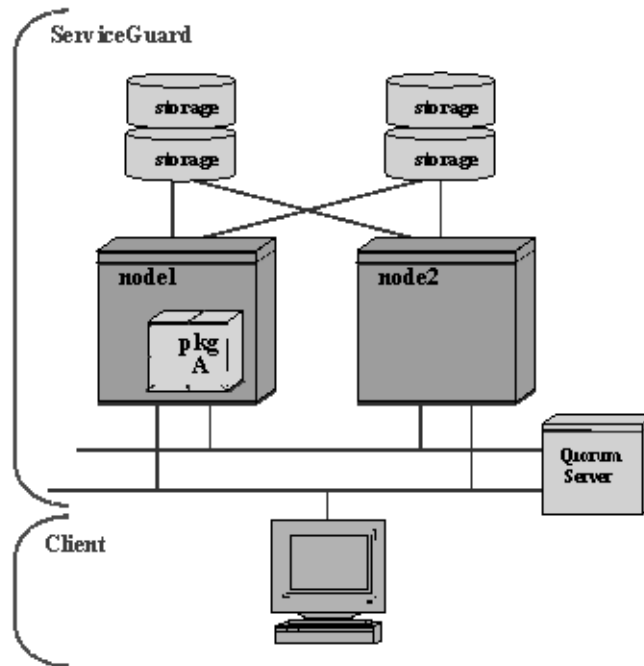


---

## Hardware Planning

Hardware planning requires examining the physical hardware itself. One useful procedure is to sketch the hardware configuration in a diagram that shows adapter cards and buses, cabling, disks and peripherals. A sample diagram for a two-node cluster is shown in Figure 4-1.

**Figure 4-1**      **Sample Cluster Configuration**



Create a similar sketch for your own cluster, and record the information on the Hardware Worksheet (page 326). Indicate which device adapters occupy which slots, and determine the bus address for each adapter. Update the details as you do the cluster configuration (described in Chapter 5).

*Use one form for each SPU.*

## SPU Information

SPU information includes the basic characteristics of the server systems you are using in the cluster. Different servers can be mixed in the same cluster however, they must all be running the same version of the operating system (i.e. IA64, x86-64, etc.) This includes both uniprocessor and multiprocessor computers.

On the worksheet, include the following items:

*Server Series Number*

Enter the series number, e.g., LXR2000.

*Host Name*

Enter the name to be used on the system as the host name.

*Memory Capacity*

Enter the memory in MB.

*Number of I/O slots*

Indicate the number of slots.

## LAN Information

While a minimum of one LAN interface per subnet is required, at least two LAN interfaces are needed to eliminate single points of network failure.

It is recommended that you configure heartbeats on all subnets, including those to be used for client data. On the worksheet, enter the following for each LAN interface:

*Subnet Name*

Enter the IP address mask for the subnet. Note that heartbeat IP addresses must be on the same subnet on each node.

*Interface Name*

Enter the name of the LAN card as used by this node to access the subnet. This name is shown by `ifconfig` after you install the card.

*IP Address*

Enter this node's host IP address intended to be used on this interface. The IP address is a string of digits separated with periods in the form 'nnn.nnn.nnn.nnn'. If the interface is a standby and does not have an IP address, enter 'Standby.'

*Kind of LAN Traffic*

Identify the purpose of the subnet. Valid types include the following:

- Heartbeat
- Client Traffic

Label the list to show the subnets that belong to a bridged net.

Information from this section of the worksheet is used in creating the subnet groupings and identifying the IP addresses in the configuration steps for the cluster manager and package manager.

## Shared Storage

SCSI may be used for up to four node clusters, or FibreChannel can be used for clusters of up to 16 nodes.

### Smart Array Cluster Storage

Smart Array cluster storage is limited to the following:

- two node cluster with MSA 500 and four node with MSA 500 G2
- the Smart Array built in interfaces or HBAs that are supported with either the MSA 500 or MSA500 G2

Check the MSA storage documentation for drive and LUN configurations that would best suit your application.

---

#### NOTE

The MSA storage and Smart Array host adapters automatically set the SCSI addresses. As a result, no user interaction is required to change or modify any of these settings.

---

### FibreChannel

FibreChannel cards can be used to connect up to 16 nodes to a disk array containing storage. After installation of the cards and the appropriate driver, the LUNs configured on the storage unit are presented to the operating system as device files, which can be used to build LVM volume groups.

---

#### NOTE

Multipath capabilities are supported by FibreChannel HBA device drivers. Check with the storage device documentation for details.

---

On the worksheet, enter the names of the device files that correspond to each LUN for the Fibre-Channel-attached storage unit.

## Disk I/O Information

This part of the worksheet lets you indicate where disk device adapters are installed. Enter the following items on the worksheet for each disk connected to each disk device adapter on the node:

*Bus Type*

Indicate the type of bus. Supported busses are F/W SCSI and FibreChannel.

*LUN Number*

Indicate the number of the LUN as defined in the storage unit.

*Slot Number*

Indicate the slot number in which the SCSI or FibreChannel interface card is inserted in the backplane of the computer.

*Address*

Enter the bus hardware path number, which is the numeric part of the host parameter, which can be seen on the system by using the following command:

```
# "cat /proc/scsi/scsi"
```

*Disk Device File*

Enter the disk device file name for each SCSI disk or LUN.

Information from this section of the worksheet is used in creating the mirrored disk configuration using LVM. In addition, it is useful to gather as much information as possible about your disk configuration.

You can obtain information about available disks by using the following commands; your system may provide other utilities as well.

- `ls /dev/cciss/c*` (Smart Array cluster storage)
- `ls /dev/hd*` (non-SCSI/FibreChannel disks)
- `ls /dev/sd*` (SCSI and FibreChannel disks)
- `du`
- `df`

## Hardware Planning

- `mount`
- `vgdisplay -v`
- `lvdisplay -v`

See the man pages on these commands for information about specific usage. The commands should be issued from *all nodes* after installing the hardware and rebooting the system. The information will be useful when doing LVM and cluster configuration.

## **Hardware Configuration Worksheet**

The Hardware configuration worksheet on page 326 will help you organize and record your specific cluster hardware configuration. Make as many copies as you need. Complete the worksheet and keep it for future reference.

## Power Supply Planning

There are two sources of power for your cluster which you will have to consider in your design: line power and uninterruptible power supplies (UPS). Loss of a power circuit should not bring down the cluster. No more than half of the nodes should be on a single power source. If a power source supplies exactly half of the nodes, it must not also supply the cluster lock disk or quorum server or the cluster will not be able to reform after a failure. See the section on cluster locks in “Cluster Configuration Planning” for more information.

To provide a high degree of availability in the event of power failure, use a separate UPS at least for each node's SPU and for the quorum server system, and ensure that disks in separate storage units are connected to different power supplies. This last rule enables disk mirroring to take place between physical disks that are connected to different power supplies as well as being on different I/O buses.

To prevent confusion, it is suggested that you label each hardware unit and power supply unit clearly with a different unit number. Indicate on the Power Supply Worksheet the specific hardware units you are using and the power supply to which they will be connected. Enter the following label information on the worksheet:

<i>Host Name</i>	Enter the host name for each SPU.
<i>Disk Unit</i>	Enter the disk drive unit number for each disk.
<i>Tape Unit</i>	Enter the tape unit number for each backup device.
<i>Other Unit</i>	Enter the number of any other unit.
<i>Power Supply</i>	Enter the power supply unit number of the UPS to which the host or other device is connected.

Be sure to follow UPS and cabinet power limits as well as SPU power limits.

## Power Supply Configuration Worksheet

The Power Supply Planning worksheet on page 327 will help you organize and record your specific power supply configuration. Make as many copies as you need. Fill out the worksheet and keep it for future reference.



## Quorum Server Planning

The quorum server (QS) provides tie-breaking services for Linux clusters. The QS is described in chapter 3 under “Cluster Quorum to Prevent Split-Brain Syndrome.”

For the quorum server, you will need a separate system that is not a part of the cluster, but which communicates with the cluster nodes on the same subnet. A quorum server:

- can be used with up to 50 clusters, not exceeding 100 nodes total.
- can support a cluster with any supported number of nodes.

Use the Quorum Server Worksheet to identify a quorum server for use with one or more clusters. You should also enter quorum server host and timing parameters on the Cluster Configuration Worksheet.

On the QS worksheet, enter the following:

*Quorum Server Host*

Enter the host name for the quorum server.

*IP Address*

Enter the IP address by which the quorum server will be accessed.

*Supported Node Names*

Enter the names of all cluster nodes that will be supported by this quorum server. These entries will be entered into `qs_authfile` on the system that is running the quorum server process.

## Quorum Server Worksheet

The Quorum Server Planning worksheet on page 328 will help you organize and record your specific quorum server hardware configuration. Make as many copies as you need. Fill out the worksheet and keep it for future reference.

## Volume Manager Planning

When designing your disk layout using LVM, you should consider the following:

- The volume groups that contain high availability applications, services, or data must be on a bus or buses available to the primary node and all adoptive nodes.
- High availability applications, services, and data should be placed in volume groups that are separate from non-high availability applications, services, and data.
- You must group high availability applications, services, and data, whose control needs to be transferred together, on a single volume group or a series of volume groups.
- You must not group two different high availability applications, services, or data, whose control needs to be transferred independently, on the same volume group.
- Your root disk must not belong to a volume group that can be activated on another node.

## Volume Groups and Physical Volume Worksheet

You can organize and record your physical disk configuration by identifying which physical disks, LUNs, or disk array groups will be used in building each volume group for use with high availability applications. Use the Volume Group and Physical Volume worksheet on page 329.

---

### NOTE

HP recommends that you use volume group names other than the default volume group names (vg01, vg02, etc.). Choosing volume group names that represent the high availability applications they are associated with (e.g., /dev/vgdatabase) will simplify cluster administration.

---

## Cluster Configuration Planning

A cluster should be designed to provide the quickest possible recovery from failures. The actual time required to recover from a failure depends on several factors:

- The length of the cluster heartbeat interval and node timeout. They should each be set as short as practical, but not shorter than 1000000 (one second) and 2000000 (two seconds), respectively; this is because of the possibility of losing heartbeat messages in many configurations. The recommended value for heartbeat interval is 1000000 (one second), and the recommended value for node timeout is within the 5 to 8 second range (5000000 to 8000000).
- The design of the run and halt instructions in the package control script. They should be written for fast execution.
- The application and database recovery time. They should be designed for the shortest recovery time.

In addition, you must provide consistency across the cluster so that:

- User names are the same on all nodes.
- UIDs are the same on all nodes.
- GIDs are the same on all nodes.
- Applications in the system area are the same on all nodes.
- System time is consistent across the cluster.
- Files that could be used by more than one node, such as `/usr` or `/opt` files, must be the same on all nodes.

## **Quorum Server Information**

The quorum server (QS) provides tie-breaking services for Linux clusters. The QS is described in chapter 3 under “Cluster Quorum to Prevent Split-Brain Syndrome.”

If you are using a quorum server, enter the following information quorum server data for later inclusion in the cluster ASCII configuration file:

- Quorum Server Host Name
- Polling Interval
- Timeout Extension Value (Optional)

## Lock LUN Information

The lock LUN is a disk-based form of tie-breaking services for Linux clusters. The lock LUN is described in chapter 3 under “Cluster Quorum to Prevent Split-Brain Syndrome.”

If you are using a lock LUN, enter the following information later in the cluster ASCII configuration file:

- Lock LUN pathnames for all cluster nodes

## Cluster Configuration Parameters

For the operation of the cluster manager, you need to define a set of cluster parameters. These are stored in the binary cluster configuration file, which is located on all nodes in the cluster. These parameters are entered by editing the cluster configuration template file created by issuing the `cmquerycl` command, as described in the chapter “Building an HA Cluster Configuration.”

The following parameters must be identified:

*CLUSTER\_NAME*     The name of the cluster as it will appear in the output of `cmviewcl` and other commands, and as it appears in the cluster configuration file.

The cluster name must not contain any of the following characters: space, slash (/), backslash (\), and asterisk (\*). All other characters are legal. The cluster name can contain up to 39 bytes.

*QS\_HOST*     The name or IP address of a host system outside the current cluster that is providing quorum server functionality. This parameter is not used if `CLUSTER_LOCK_LUN` is used. The `QS_HOST` name can contain up to 39 bytes.

*QS\_POLLING\_INTERVAL*

The time (in microseconds) between attempts to contact the quorum server to make sure it is running. Default is 300,000,000 microseconds (5 minutes). This parameter is not used if `CLUSTER_LOCK_LUN` is used.

*QS\_TIMEOUT\_EXTENSION*

The quorum server timeout is the time during which the quorum server is not communicating with the cluster. After this time, the cluster will mark the quorum server `DOWN`. This time is calculated based on Serviceguard parameters, but you can increase it by adding an additional number of microseconds as an extension.

The `QS_TIMEOUT_EXTENSION` is an optional parameter. A value higher than `NODE_TIMEOUT` will increase the failover time. This parameter is not used if `CLUSTER_LOCK_LUN` is used.

*NODE\_NAME*

The hostname of each system that will be a node in the cluster. A cluster may contain up to 16 nodes. The node name can contain up to 31 bytes.

*CLUSTER\_LOCK\_LUN*

The pathname of the device file to be used for the lock LUN on the previously specified node. This parameter is not used if `QS_HOST` is used. The pathname can contain up to 255 characters.

*NETWORK\_INTERFACE*

The name of each LAN that will be used for heartbeats or for user data. An example is 'eth0'.

*HEARTBEAT\_IP*

IP notation indicating the subnet that will carry the cluster heartbeat. Note that heartbeat IP addresses must be on the same subnet on each node.

---

**NOTE**

The use of a private heartbeat network is not advisable if you plan to use Remote Procedure Call (RPC) protocols and services. RPC assumes that each network adapter device or I/O card is connected to a route-able network. An isolated or private heartbeat LAN is not route-able, and could cause an RPC request-reply, directed to that LAN, to risk time-out without being serviced.

NFS, NIS and NIS+, and CDE are examples of RPC based applications that are frequently used on UNIX systems. Other third party and home-grown applications may also use RPC services directly through the RPC API libraries. If necessary, consult with the application vendor to confirm its usage of RPC.

---

*STATIONARY\_IP*

The IP address of each monitored subnet that does not carry the cluster heartbeat. You can identify any number of subnets to be monitored. If you want to separate application data from heartbeat messages, define a monitored non-heartbeat subnet here.

*HEARTBEAT\_INTERVAL*

The normal interval between the transmission of heartbeat messages from one node to the other in the cluster. Enter a number of microseconds. The default value is 1,000,000 microseconds; setting the parameter to a value less than the default is *not recommended*.

The default should be used where possible. The maximum value recommended is 15 seconds, and the maximum value supported is 30 seconds. This value should be at least half the value of *NODE\_TIMEOUT* (below).

*NODE\_TIMEOUT*

The time after which a node may decide that the other node has become unavailable and initiate cluster reformation. Enter a number of microseconds. The default value is 2,000,000 microseconds. Minimum is 2 \* (Heartbeat Interval). The maximum recommended value for this parameter is 30,000,000. The default setting yields the fastest cluster reformations. However, the user of the default value increases the potential for spurious reformations due to momentary system hangs or network load spikes. For a significant portion of installations, a setting of 5,000,000 to 8,000,000 (5 to 8 seconds) is more appropriate.

The maximum value recommended is 30 seconds and the maximum value supported is 60 seconds.

*AUTO\_START\_TIMEOUT*

The amount of time a node waits before it stops trying to join a cluster during automatic cluster startup. All nodes wait this amount of time for other nodes to begin startup before the cluster completes the operation. The time should be selected based on the slowest boot time in the cluster. Enter a value equal to the boot time of the slowest booting node minus the boot time of the fastest booting node plus 600 seconds (ten minutes).

Default is 600,000,000 microseconds.



#### *NETWORK\_POLLING\_INTERVAL*

The frequency at which the networks configured for Serviceguard are checked. In the ASCII cluster configuration file, this parameter is `NETWORK_POLLING_INTERVAL`.

Default is 2,000,000 microseconds in the ASCII file. Thus every 2 seconds, the network manager polls each network interface to make sure it can still send and receive information. Changing this value can affect how quickly a network failure is detected.

The minimum value is 1,000,000 (1 second). The maximum value recommended is 15 seconds, and the maximum value supported is 30 seconds.

#### *ACCESS CONTROL POLICIES*

Specify three things for each policy: `USER_NAME`, `USER_HOST`, and `USER_ROLE`. For Serviceguard Manager, `USER_HOST` must be the name of the Session node. Policies set in the configuration file of a cluster and its packages must not be conflicting or redundant. For more information, see “Username Validation” on page 126.

#### *MAX\_CONFIGURED\_PACKAGES*

This parameter sets the maximum number of packages that can be configured in the cluster. Default is 0, which means that you must set this parameter if you want to use packages. The minimum value is 0, and the maximum value is 150.

Set this parameter to a value that is high enough to accommodate a reasonable amount of future package additions without the need to bring down the cluster to reset the parameter. However, be sure not to set the parameter so high that memory is wasted. The use of packages requires 6MB plus about 80 KB of lockable memory on all cluster nodes.

## **Cluster Configuration Worksheet**

The following worksheet on page 332 will help you to organize and record your cluster configuration. Make as many copies as you need. Complete and save this worksheet for future reference.

---

## Package Configuration Planning

Planning for packages involves assembling information about each group of highly available services. Some of this information is used in creating the package configuration file, and some is used for editing the package control script.

---

### NOTE

Volume groups that are to be activated by packages must also be defined as cluster aware in the cluster configuration file. See the previous section on “Cluster Configuration Planning.”

---

## Logical Volume and File System Planning

You may need to use logical volumes in volume groups as part of the infrastructure for package operations on a cluster. When the package moves from one node to another, it must be able to access data residing on the same disk as on the previous node. This is accomplished by activating the volume group and mounting the file system that resides on it.

In Serviceguard, high availability applications, services, and data are located in volume groups that are on a shared bus. When a node fails, the volume groups containing the applications, services, and data of the failed node are deactivated on the failed node and activated on the adoptive node. In order to do this, you have to configure the volume groups so that they can be transferred from the failed node to the adoptive node.

---

### NOTE

To prevent an operator from accidentally activating volume groups on other nodes in the cluster, versions A.11.16.07 and later of Serviceguard for Linux include a type of VG activation protection. This is based on the “hosttags” feature of LVM2 and so is not available on RedHat 3. This feature is not mandatory, but HP strongly recommends you implement it as you upgrade existing clusters and create new ones.

---

As part of planning, you need to decide the following:

- What volume groups are needed?
- How much disk space is required, and how should this be allocated in logical volumes?
- What file systems need to be mounted for each package?
- Which nodes need to import which logical volume configurations.
- If a package moves to an adoptive node, what effect will its presence have on performance?

Create a list by package of volume groups, logical volumes, and file systems. Indicate which nodes need to have access to common file systems at different times.

It is recommended that you use customized logical volume names that are different from the default logical volume names (lv01, lv02, etc.). Choosing logical volume names that represent the high availability applications that they are associated with (for example, lvoldatabase) will simplify cluster administration.

```
# /dev/vg01/lvoldb1 /applic1 ext2 defaults 0 1 # These six entries are
# /dev/vg01/lvoldb2 /applic2 ext2 defaults 0 1 # for information purposes
# /dev/vg01/lvoldb3 raw_tables ignore ignore 0 0 # only. They record the
# /dev/vg01/lvoldb4 /general ext2 defaults 0 2 # logical volumes that
# /dev/vg01/lvoldb5 raw_free ignore ignore 0 0 # exist for Serviceguard's
# /dev/vg01/lvoldb6 raw_free ignore ignore 0 0 # HA package. Do not uncomment.
```

To further document your package-related volume groups, logical volumes, and file systems on each node, you can add *commented* lines to the `/etc/fstab` file. The following is an example for a database application:

Create an entry for each logical volume, indicating its use for a file system or for a raw device.

---

## CAUTION

---

Do *not* use `/etc/fstab` to mount file systems that are used by Serviceguard packages.

Details about creating, exporting, and importing volume groups in Serviceguard are given in the chapter on “Building an HA Cluster Configuration.”

## Planning for Expansion

You can add packages to a running cluster. This process is described in the chapter “Cluster and Package Administration.” When adding packages, be sure not to exceed the value of `MAX_CONFIGURED_PACKAGES` as defined in the cluster configuration file.

When planning on increasing the number of packages, remember that the use of packages requires 6MB plus about 80KB per package of lockable memory on each cluster node.

## Choosing Switching and Failover Behavior

To determine failover behavior, you can define a package failover policy that governs which nodes will automatically start up a package that is not running. In addition, you can define a fallback policy that determines whether a package will be automatically returned to its primary node when that is possible.

Table 3-3 in the previous chapter describes different types of failover behavior and settings of parameters in the ASCII package configuration file that determine each behavior.

## Package Configuration File Parameters

Prior to generation of the package configuration file, assemble the following package configuration data. The following parameters must be identified and entered on the worksheet *for each package*:

*PACKAGE\_NAME*

The name of the package. The package name must be unique in the cluster. It is used to start, stop, modify, and view the package. The package name can be from 1 to 39 bytes. It must not contain any of the following illegal characters: space, slash (/), backslash(\), and asterisk (\*). All other characters are legal.

*PACKAGE\_TYPE*

The type of the package. This parameter indicates valid type is `FAILOVER`.

*FAILOVER\_POLICY*

The policy used by the package manager to select the node on which the package should be automatically started.

Default is `CONFIGURED_NODE`, which means the next available node in the list of node names for the package. The order of node name entries dictates the order of preference when selecting the node. The alternate policy is `MIN_PACKAGE_NODE`, which means the node from the list that is running the fewest other packages at the time this package is to be started.

*FAILBACK\_POLICY*

The policy used to determine what action the package manager should take if the package is not running on its primary node and its primary node is capable of running the package.

Default is `MANUAL`, which means no attempt will be made to move the package back to its primary node when it is running on an alternate node. The alternate policy is `AUTOMATIC`, which means that the package will be halted and restarted on its primary node as soon as the primary node is capable of running the package and, if `MIN_PACKAGE_NODE` has been selected as the Package Failover Policy, the primary node is now running fewer packages than the current node.

*NODE\_NAME*

The names of primary and alternate nodes for the package. Enter a node name for each node on which the package can run.

The order in which you specify the node names is important. First list the primary node name, then the first adoptive node name, then the second adoptive node name, followed, in order, by additional node names. In a failover, control of the package will be transferred to the next adoptive node name listed in the package configuration file.

The node name can contain up to 31 bytes.

*AUTO\_RUN*

This parameter determines how a package may start up. If `AUTO_RUN` is set to `YES`, the package will start up automatically on an eligible node if one is available,

and will be able to fail over automatically to another node. If `AUTO_RUN` is set to `NO`, the package will not start up automatically when the cluster starts running, and will not be able to fail over automatically to another node.

Default is `YES`, which allows a package to start up normally on an available node. Enter `YES` or `NO` in the package ASCII file.

---

**NOTE**

---

Serviceguard disables `AUTO_RUN` automatically whenever the command `cmhalt pkg` is executed. The user needs to re-enable `AUTO_RUN` using the `cmmodpkg -e <pkg>` command.

#### *NODE\_FAIL\_FAST\_ENABLED*

In the event of the failure of the control script itself or the failure of a subnet, if this parameter is set to `YES`, Serviceguard will issue a TOC on the node where the control script fails.

Enter `YES` or `NO`. The default is `NO`.

If `NODE_FAIL_FAST_ENABLED` is set to `YES`, the node where the package is running will be halted if one of the following failures occurs:

- A package subnet fails and no backup network is available
- The halt script does not exist
- Serviceguard is unable to execute the halt script
- The halt script times out

However, if the package halt script fails with “exit 1”, Serviceguard does not halt the node, but sets `NO_RESTART` for the package, which causes package switching (`AUTO_RUN`) to be disabled, thereby preventing the package from starting on any adoptive node.

#### *RUN\_SCRIPT* and *HALT\_SCRIPT*

This run script contains package run instructions and the halt script contains package halt instructions. It is recommended that you use the same script as both the run and halt script. If you wish to separate the package run instructions and package halt instructions into separate scripts, the package configuration file allows you to do this by naming two separate scripts.

However, under most conditions, it is simpler to use the name of a single control script as the name of the `RUN_SCRIPT` and the `HALT_SCRIPT`. When the package starts, its run script is executed and passed the parameter 'start'; similarly, at package halt time, the halt script is executed and passed the parameter 'stop'.

Enter the full pathname of the package control script. (The script must reside in a directory that contains the string "cmcluster.") Ensure that this script is executable.

If you choose to write separate package run and halt scripts, be sure to include identical configuration information (such as node names, IP addresses, etc.) in both scripts.

#### *`RUN_SCRIPT_TIMEOUT` and `HALT_SCRIPT_TIMEOUT`*

If the script has not completed by the specified timeout value, Serviceguard will terminate the script. Enter a value in seconds.

The default is 0, or no timeout. The minimum is 10 seconds, but the minimum `HALT_SCRIPT_TIMEOUT` value must be greater than the sum of all the Service Halt Timeout values. The absolute maximum value is restricted only by the UNIX parameter `ULONG_MAX`, for an absolute limit of 4,294 seconds.

If the timeout is exceeded:

- Control of the package will not be transferred.
- The run or halt instructions will not be run.
- Global switching will be disabled.
- The current node will be disabled from running the package.



- The control script will exit with status 1.

If the halt script timeout occurs, you may need to perform manual cleanup. See “Package Control Script Hangs or Failures” in Chapter 8.

*SERVICE\_NAME*

Enter a unique name for each service. Define one *SERVICE\_NAME* entry for each service. You can configure a maximum of 30 services per package. The service name must not contain any of the following illegal characters: space, slash (/), backslash (\), and asterisk (\*). All other characters are legal. The service name can contain up to 39 bytes.

*SERVICE\_FAIL\_FAST\_ENABLED*

Enter YES or NO for each service. This parameter indicates whether or not the failure of a service results in the failure of a node. If the parameter is set to Enabled, in the event of a service failure, Serviceguard will halt the node on which the service is running with a TOC. (An attempt is first made to reboot the node prior to the TOC.) The default is NO. Define one *SERVICE\_FAIL\_FAST\_ENABLED* entry for each service.

*SERVICE\_HALT\_TIMEOUT*

In the event of a service halt, Serviceguard will first send out a SIGTERM signal to terminate the service. If the process is not terminated, Serviceguard will wait for the specified timeout before sending out the SIGKILL signal to force process termination. Define one *SERVICE\_HALT\_TIMEOUT* entry for each service.

Default is 300 seconds (5 minutes). The minimum is 1 second, and the maximum value is restricted only by the UNIX parameter *ULONG\_MAX*, for an absolute limit of 4,294 seconds.

Define one *SERVICE\_HALT\_TIMEOUT* entry for each service.

*SUBNET*

Enter the IP subnets that are to be monitored for the package, one per line.

*ACCESS CONTROL POLICIES*

Specify three things for each policy: `USER_NAME`, `USER_HOST`, and `USER_ROLE`. For Serviceguard Manager, `USER_HOST` must be the name of the Session node. Policies set in the configuration file of a cluster and its packages must not be conflicting or redundant. For more information, see “Username Validation” on page 126.

**Package Control Script Variables**

The control script that accompanies each package must also be edited to assign values to a set of variables. Set variables, or leave them unset, as described below.

<i>PATH</i>	Specify the path to be used by the script.
<i>DATA_REP</i>	This parameter indicates whether the script is being used in a remote data replication scheme. Do not set it unless the remote data replication scheme you are using requires. The default is “none.”
<i>RAIDTAB</i>	<i>Do not use this parameter; it is deprecated in Serviceguard for Linux.</i>  (If used, <i>RAIDTAB</i> specifies the configuration file that contains the names of the multiple device groups used by the package for multiple links to a disk array.)
<i>RAIDSTART and RAIDSTOP</i>	<i>Do not use this parameter; it is deprecated in Serviceguard for Linux.</i> (If used, <i>RAIDSTART</i> and <i>RAIDSTOP</i> specify the method of activation for md.)
<i>VGCHANGE</i>	This variable specifies the LVM activation command used by the script. Leave the default ( <code>VGCHANGE=“vgchange -a y”</code> ) if you want volume groups activated in the default mode.

*Volume Groups (VG)*

These parameters specify the names of LVM volume groups that are activated by the package. In the package control script file, this variables is an array: `VG[ ]`.

On starting the package, the script may activate one or more volume groups, and at halt time, the script deactivates each volume group. All volume groups must be accessible on each target node.

Include as many volume groups (VGs) as needed. If you are using raw devices, the `LV`, `FS`, and `FS_MOUNT_OPT` entries are not needed. Only cluster aware volume groups should be specified in package control scripts. To make a volume group cluster aware, enter it as part of the cluster configuration. See above, “Cluster Configuration Planning.”

<i>MD</i>	<i>Do not use this parameter; it is deprecated in Serviceguard for Linux.</i> (If used, the <i>MD</i> array stores the names of multiple devices that are employed for storage in the package and defined in the RAIDTAB file.)
<i>LV</i>	The name of a logical volume hosting a file system that will be mounted by the package.
<i>FS</i>	The name of the mount point for a file system to be mounted by the package.
<i>FS_TYPE</i>	<p>The file system type of a file system being mounted by the package. Three types of file systems are supported:</p> <ul style="list-style-type: none"><li>• <code>ext2</code>, the basic Linux file system</li><li>• <code>ext3</code>, a journaling file system that improves performance when the file system is fsck'd.</li><li>• <code>reiserfs</code>, a journaling file system with B-tree data storage.</li></ul> <p><code>ext3</code> is the default for Red Hat 3 and Red Hat 4. <code>reiserfs</code> is the default for SLES9.</p>
<i>FS_MOUNT_OPT</i>	The options that should be used with the <code>mount</code> command when issued from the package control script during package startup.
<i>FS_UMOUNT_OPT</i>	The options that should be used with the <code>umount</code> command when issued from the package control script during package shutdown.

*FS\_UNMOUNT\_COUNT*

The number of unmount attempts allowed for each file system during package shutdown. Default is 1.

*FS\_MOUNT\_RETRY\_COUNT*

The number of mount retries for each file system. The default is 0. During startup, if a mount point is busy and *FS\_MOUNT\_RETRY\_COUNT* is 0, package startup will fail and the script will exit with 1. If a mount point is busy and *FS\_MOUNT\_RETRY\_COUNT* is greater than 0, the script will attempt to kill the user responsible for the busy mount point for the number of times specified in *FS\_MOUNT\_RETRY\_COUNT*. If the mount still fails after this number of attempts, the script will exit with 1.

*CONCURRENT\_FSCK\_OPERATIONS*

Specifies the number of concurrent fsck commands to allow during package startup. The default is 1. Setting this variable to a higher value may improve performance when checking a large number of file systems. If a value less than 1 is specified, the script defaults the variable to 1 and writes a warning message in the package control script log file.

*CONCURRENT\_MOUNT\_AND\_UNMOUNT\_OPERATIONS*

Specifies the number of mounts and umounts to allow during package startup or shutdown. The default is 1. Setting this variable to a higher value may improve performance while mounting or unmounting a large number of file systems. If a value less than 1 is specified, the script defaults the variable to 1 and writes a warning message in the package control script log file.

*IP Addresses and SUBNETs*

These are the IP addresses by which a package is mapped to a LAN card. Indicate the IP addresses and subnets for each IP address you want to add to an interface card.

In the package control script file, these variables are entered in pairs. Example `IP[0]=192.10.25.12` and `SUBNET[0]=192.10.25.0`. (In this case the subnet mask is 255.255.255.0.)

#### *HA\_APP\_SERVER*

Specify that High Availability dependent servers are being used with this package. One of the key servers for HA is Network File System (NFS). You need to set `HA_APP_SERVER` to yes in order to let the control script verify, start, and stop the NFS daemons. This parameter is for use only with certain HP supported toolkits. The toolkits that currently use this parameter (either `pre_IP` or `post_IP`) include NFS, SAMBA, and Apache. The default is to leave this parameter commented out.

#### *Service Name*

Enter a unique name for each specific service within the package. All services are monitored by Serviceguard. You may specify up to 30 services per package. Each name must be unique within the cluster. The service name is the name used by `cmrunserv` and `cmhaltserv` inside the package control script. It must be the same as the name specified for the service in the package ASCII configuration file.

In the package control script file, enter values into an array known as `SERVICE_NAME`. Enter one service name for each service. The `SERVICE_NAME`, `SERVICE_CMD`, and `SERVICE_RESTART` parameters are set in the package control script in groups of three.

The service name must not contain any of the following illegal characters: space, slash (/), backslash (\), and asterisk (\*). All other characters are legal. The service name can contain up to 39 bytes.

#### *Service Command*

For each named service, enter a Service Command. This command will be executed through the control script by means of the `cmrunserv` command.

In the package control script file, enter values into an array known as `SERVICE_CMD`. Enter one service command string for each service. The `SERVICE_NAME`, `SERVICE_CMD`, and `SERVICE_RESTART` parameters are set in the package control script in groups of three.

#### *Service Restart Parameter*

Enter a number of restarts. One valid form of the parameter is `-r n` where `n` is a number of retries. A value of `-r 0` indicates no retries. A value of `" "` means do not restart and `-R` indicates infinite restarts. The default is 0, or no restarts.

In the package control script file, enter values into an array known as `SERVICE_RESTART`. Enter one restart value for each service. The `SERVICE_NAME`, `SERVICE_CMD`, and `SERVICE_RESTART` parameters are set in the package control script in groups of three.

The package control script will clean up the environment and undo the operations in the event of an error. Refer to the section on “How Package Control Scripts Work” in Chapter 3 for more information.

## Package Configuration Worksheet

Assemble your package configuration and control script data in a separate worksheet for each package. Examples of each are shown in Figure 4-2 and Figure 4-3.

**Figure 4-2**

### Package Configuration Worksheet

```
=====
Package Configuration File Data:
=====

Package Name: _____pkg11_____

Failover Policy: _CONFIGURED_NODE__   Failback Policy: __AUTOMATIC__

Primary Node: _____ftsys9_____

First Failover Node:___ftsys10_____

Additional Failover Nodes:_____

Package Run Script: __/usr/local/cmcluster/pkg1/control.sh
```

```
Timeout: _NO_TIMEOUT_  
  
Package Halt Script: __/usr/local/cmcluster/pkg1/control.sh_  
  
Timeout: _NO_TIMEOUT_  
  
Package AutoRun Enabled? __YES__ Local LAN Failover Allowed? __YES__  
  
Node Failfast Enabled?      ____NO____  
Access Policies  
  
User: __ ANY_USER  
Host: __ ftsys9__  
Role: __ full_admin__  
  
User: __ sara itgrp lee __  
Host: __ ftsys10__  
Role: __ package_admin__
```

**Figure 4-3**      **Package Control Script Worksheet**

```
=====
PACKAGE CONTROL SCRIPT WORKSHEET                                     Page ____ of ____
=====
Package Control Script Data:
=====

PATH_____

VGCHANGE_____

VG[0]___/dev/gv01___LV[0]___/dev/vg01/1v011___FS[0]___/mnt1_____

VG[1]_____LV[1]_____FS[1]_____

VG[2]_____LV[2]_____FS[2]_____

FS Umount Count: _____FS Mount Retry Count:_____

IP[0] ___15.13.171.14_____SUBNET ___15.13.168.0_____

IP[1] _____SUBNET _____

Service Name: __svc1___Command: ___/usr/bin/MySvc -f__Restart: _-r 2__

Service Name: _____Command: _____Restart: _____
```

---

**NOTE**

---

Please read and follow the note about the MD driver in the latest Package Control Script (Chapter 6).



# 5

## Building an HA Cluster Configuration

This chapter and the next take you through the configuration tasks required to set up a Serviceguard cluster. These procedures are carried out on one node, called the **configuration node**, and the resulting binary file is distributed by Serviceguard to all the nodes in the cluster. In the examples in this chapter, the configuration node is named `ftsys9`, and the sample target node is called `ftsys10`. This chapter describes the following cluster configuration tasks:

- Understanding the Location of Serviceguard Files
- Preparing Your Systems
- Setting up the Quorum Server
- Setting up the Lock LUN
- Creating the Logical Volume Infrastructure
- Configuring the Cluster
- Managing the Running Cluster

Configuring packages is described in the next chapter.

Use the Serviceguard man pages for each command to obtain information about syntax and usage.

## Preparing Your Systems

Before configuring your cluster, ensure that all cluster nodes possess the appropriate security files, kernel configuration and NTP (network time protocol) configuration.

### Understanding the Location of Serviceguard Files

Serviceguard uses a special file, `cmcluster.conf`, to define the locations for configuration and log files within the Linux file system. The different distributions may use different locations. The following are example locations for a Red Hat distribution:

```
##### cmcluster.conf #####
#
# Highly Available Cluster file locations
#
# This file must not be edited
#####
SGROOT=/usr/local/cmclusters      # SG root directory
SGCONF=/usr/local/cmcluster/conf  # configuration files
SGSBIN=/usr/local/cmcluster/bin   # binaries
SGLBIN=/usr/local/cmcluster/bin   # binaries
SGLIB=/usr/local/cmcluster/lib    # libraries
SGRUN=/usr/local/cmcluster/run    # location of core dumps from daemons
SGAUTOSTART=/usr/local/cmcluster/conf/cmcluster.rc # SG Autostart file
```

The following are example locations for a SuSE distribution:

```
##### cmcluster.conf #####
#
# Highly Available Cluster file locations
#
# This file must not be edited
#####
SGROOT=/opt/cmcluster            # SG root directory
SGCONF=/opt/cmcluster/conf       # configuration files
SGSBIN=/opt/cmcluster/bin        # binaries
SGLBIN=/opt/cmcluster/bin        # binaries
SGLIB=/opt/cmcluster/lib         # libraries
SGRUN=/opt/cmcluster/run         # location of core dumps from daemons
SGAUTOSTART=/opt/cmcluster/conf/cmcluster.rc # SG Autostart file
```

Throughout this document, system filenames are usually given with one of these location prefixes. Thus, references to `$SGCONF/<FileName>` can be resolved by supplying the definition of the prefix that is found in this

file. For example, if `SGCONF` is `/usr/local/cmcluster/conf`, then the complete pathname for file `$SGCONF/cmclconfig` would be `/usr/local/cmcluster/conf/cmclconfig`.

## Enabling Serviceguard Command Access

To allow the creation of a Serviceguard configuration, you should complete the following steps *on all cluster nodes* before running any Serviceguard commands:

1. Make sure the root user's path includes the Serviceguard executables. This can be done by adding the following environment variable definition to the root user's profile for Red Hat:

```
PATH=$PATH:/usr/local/cmcluster/bin
```

For SuSE:

```
PATH=$PATH:/opt/cmcluster/bin
```

2. Edit the `/etc/man.config` file to include the following line for Red Hat:

```
MANPATH /usr/local/cmcluster/doc/man
```

For SUSE:

```
MANPATH /opt/cmcluster/doc/man
```

This will allow use of the Serviceguard man pages.

3. Enable use of Serviceguard variables.

If the Serviceguard variables are not defined on your system, then include the file `/etc/cmcluster.conf` in your login profile for user root.

```
# . /etc/cmcluster.conf
```

When this command completes, the prompt appears. You can confirm the access to the one of the variables as follows:

```
# cd $SGCONF
```

## Editing Security Files

Serviceguard daemons grant access to commands by matching incoming hostname and username against defined access control policies. To understand how to properly configure these policies, administrators need to understand how Serviceguard handles hostnames, IP addresses, usernames and the relevant configuration files.

For redundancy, Serviceguard utilizes all available IPv4 networks for communication. If a Serviceguard node is able to communicate with another node on that interface, the access control policy needs to include the primary IP address for that interface.

### IP Address Resolution

Access control policies for Serviceguard are name-based. IP addresses for incoming connections must be resolved into hostnames to match against access control policies.

Communication between two Serviceguard nodes could be received over any of their shared networks. Therefore, all of their primary addresses on each of those networks needs to be identified.

Serviceguard supports using aliases. An IP address may resolve into multiple hostnames, one of those should match the name defined in the policy.

### Configuring IP Address Resolution

Serviceguard uses the operating system's built in name resolution services. It is recommended that name resolutions are defined in the node's `/etc/hosts` file first rather than rely on DNS or NIS services for the proper functioning of the cluster.

For example, consider a two node cluster (gryf and sly) with two private subnets and a public subnet. They will be granting permission to a non-cluster node (bit) who does not share the private subnets. The `/etc/hosts` file on both cluster nodes should contain:

```
15.145.162.131 gryf.uksr.hp.com gryf
10.8.0.131 gryf.uksr.hp.com gryf
10.8.1.131 gryf.uksr.hp.com gryf
15.145.162.132 sly.uksr.hp.com sly
10.8.0.132 sly.uksr.hp.com sly
```

```
10.8.1.132 sly.uksr.hp.com sly
15.145.162.150 bit.uksr.hp.com bit
```

---

**NOTE**

Serviceguard will only recognize the hostname in a fully qualified domain name (FQDN). For example, two nodes `gryf.uksr.hp.com` and `gryf.cup.hp.com` could not be in the same cluster as they would both be treated as the same host `gryf`.

---

Serviceguard also supports domain name aliases. If other applications require different interfaces to have a unique primary hostname, the Serviceguard hostname can be one of the aliases. For example:

```
15.145.162.131 gryf.uksr.hp.com gryf node1
10.8.0.131 gryf.uksr.hp.com gryf
10.8.1.131 gryf.uksr.hp.com gryf
15.145.162.132 sly.uksr.hp.com sly node2
10.8.0.132 sly.uksr.hp.com sly
10.8.1.132 sly.uksr.hp.com sly
```

In this configuration, the private subnets' primary name is unique. By providing the alias, Serviceguard can still associate this IP address with the proper node and match it in a access control policy.

The name service switch policy should be configured to consult the `/etc/hosts` file before other sources such as DNS, NIS, or LDAP. Ensure that the `/etc/nsswitch.conf` file on all the cluster nodes lists 'files' first, then followed by other services. For example:

For DNS, enter: (one line):

```
hosts: files [NOTFOUND=continue UNAVAIL=continue] dns
[NOTFOUND=return UNAVAIL=return]
```

For NIS, enter: (one line):

```
hosts: files [NOTFOUND=continue UNAVAIL=continue] nis
[NOTFOUND=return UNAVAIL=return]
```

## Username Validation

Serviceguard relies on the `ident` service of the client node to verify the username of the incoming network connection. If the Serviceguard daemon is unable to connect to the client's `ident` daemon, permission will be denied.

Root on a node is defined as any user who has the UID of 0. For a user to be identified as root on a remote system, the “root” user entry in `/etc/passwd` for the local system must come before any other user who may also be UID 0. The `ident` daemon will return the username for the first UID match. For Serviceguard to consider a remote user as a root user on that remote node, the `ident` service must return the username as “root”.

It is possible to configure Serviceguard to not use the `ident` service, however this configuration is not recommended. Consult the `ratepayer` “Securing Serviceguard” for more information.

To disable the Serviceguard features, do the following steps on each node after installing Serviceguard A.11.16.01 but before having each node re-join the cluster (e.g. before issuing a `cmrunnode` or `cmruncl`).

For Red Hat and SUSE:

1. Change the `server_args` parameter in the file `/etc/xinetd.d/hacl-cfg`  
from:  
`server_args = -c`  
to  
`server_args = -c -i`
2. Change the `server_args` parameter in the `/etc/xinetd.d/hacl-probe` file to include the `-i`

For SUSE this would be changed from:

```
server_args = -f /opt/cmom/log/cmomd.log -r /opt/cmom/run  
to  
server_args = -i -f /opt/cmom/log/cmomd.log -r  
/opt/cmom/run
```

For Red Hat this would be changed from:

```
server_args = -f /user/local/cmom/log/cmomd.log -r  
/user/local/cmom/run
```

```
to  
server_args = -i -f /user/local/cmom/log/cmomd.log -r  
/user/local/cmom/run
```

3. Restart xinetd: `/etc/init.d/xinetd restart`

## Access Roles

Serviceguard has two levels of access:

- **Root Access:** Users who have been authorized for root access have total control over the configuration of the cluster and packages.
- **Non-root Access:** Non-root users can be assigned one of four roles:
  - **Monitor:** These users have read-only access to the cluster and its packages. Command line users can issue these commands: `cmviewcl`, `cmquerycl`, `cmgetconf`, and `cmviewconf`. Serviceguard Manager users can see status and configuration information on the map, tree and properties.
  - **(one) Package Admin:** Applies only to a specific package. On the command line, these users can issue the commands for the specified package: `cmrunpkg`, `cmhaltpkg`, `cmmodnet`, `cmrunserv`, `cmhaltserv`, `cmstartres`, `cmstopres`, and `cmmodpkg`. Serviceguard Manager users can see these Admin menu options for their specific package: Run Package, Halt Package, Move Package, and Enable or Disable Switching. Package admins can not configure or create packages. Package Admin includes the privileges of the Monitor role.
  - **(all) Package Admin:** Applies to all packages in the cluster. The commands are the same as the role above. Package Admin includes the privileges of the Monitor role.
  - **Full Admin:** These users can administer the cluster. On the command line, these users can issue these commands in their cluster: `cmruncl`, `cmhaltcl`, `cmrunnode`, and `cmhaltnode`. Full Admins can not configure or create a cluster. In the Serviceguard Manager, they can see the Admin menu for their cluster and any packages in their cluster. Full Admin includes the privileges of the Package Admin role.

Setting access control policies uses different mechanisms depending on the state of the node. Nodes not configured into a cluster use different security configurations than nodes in a cluster. The following two sections discuss how to configure these access control policies.

### Setting Controls for an Unconfigured Node

Serviceguard access control policies define what a remote node can do to the local node. A new install of Serviceguard will not have any access control policies defined. To enable this node to be included in a cluster, a policy must be defined to allow access for root from the other potential cluster nodes. For Serviceguard Manager, policies must be defined to allow remote COM servers to Monitor or configure the node. These policies will only be in effect while a node is not configured into a cluster.

Unconfigured nodes may authorize two levels of access to remote users: root and non-root. Users with root access may use any cluster configuration commands. Users with non-root access are assigned the Monitor role giving them read-only access to the nodes configuration.

When a Serviceguard node is not configured in a cluster it relies on one of two possible security mechanisms for authorizing remote users:

- If the file `$SGCONF/cmclnodelist` file exists, Serviceguard will use its contents to authorize remote users.
- The host equivalency files used by `r-commands`, `~/.rhosts` and `/etc/hosts.equiv` (`hostsequiv`).

The use of `cmclnodelist` is strongly recommended.

Serviceguard will check for the existence of `$SGCONF/cmclnodelist` before attempting to access `hostsequiv`. If the file exists, Serviceguard will not check other authorization mechanisms. With regard to Serviceguard, using either `cmclnodelist` or `hostsequiv` provides the same levels of security. Administrators may choose to use `cmclnodelist` file instead of `hostsequiv` in installations which may wish to limit `r-command` access.

For backwards compatibility, a node in an unconfigured state may define access control policies based on IP address. The primary IP address on each interface Serviceguard uses for communication must have it's own policy if name services are not configured as specified above. Once a node is configured into a cluster, IP addresses can no longer be used for these policies.



## Using the cmclnodelist File

The `cmclnodelist` file is not created by default in new installations. If administrators want to create this bootstrap file they should add a comment such as the following:

```
#####  
# Do Not Edit This File  
# This is only a temporary file to bootstrap an unconfigured  
# node with Serviceguard version A.11.16  
# Once a cluster is created, Serviceguard will not consult  
# this file.
```

```
#####
```

The format for entries in the `cmclnodelist` file is as follows:

```
[hostname or IP address] [user] [#Comment]
```

For example:

```
gryf root # Cluster 1,Node 1  
gryf user1 # Cluster 1, Node 1  
sly root # Cluster 1, Node 2  
sly user1 # Cluster 1, Node 2  
bit root # Administration /COM Server
```

In this example, `root` on the nodes `gryf`, `sly`, and `bit` all have root access to the node with this file. The non-root user “`user1`” has the Monitor role from nodes `gryf` and `sly`.

Serviceguard also accepts the use of a “+” in the `cmclnodelist` file which indicates that any root user on any node may configure this node and any non-root user has the Monitor role.

## Using Equivalent Hosts

For installations that wish to use `hostsequiv`, the primary IP addresses or hostnames for each node in the cluster needs to be authorized. For more information on using `hostsequiv`, see `man hosts.equiv(4)` or the HP-UX guide, “Managing Systems and Workgroups”.

Though `hostsequiv` allows defining any user on any node as equivalent to root, Serviceguard will not grant root access to any user who is not root on the remote node. Such a configuration would grant non-root access to that user.

## Setting Access Controls for a Configured Clusters

Once nodes are configured in a cluster, different cluster wide security mechanisms are used. Changes to `cmclnodelist` file and `hostsequiv` are ignored. Root users within the cluster are automatically granted root access. All other users may be optionally authorized for non-root roles. Root access cannot be given to root users on nodes outside the cluster.

Access control policies for a configured cluster are defined in the `ascii` cluster configuration file. Access control policies for a specific package are defined in the package configuration file. Any combination of hosts and users may be assigned roles for the cluster. You can have up to 200 access policies defined for a cluster.

Access policies are defined by three parameters in the configuration file:

- `USER_NAME` can either be `ANY_USER`, or a maximum of 8 login names from the `/etc/passwd` file on user host.
- `USER_HOST` is where the user can issue Serviceguard commands. If using Serviceguard Manager, it is the COM server. Choose one of these three values: `ANY_SERVICEGUARD_NODE`, or (any) `CLUSTER_MEMBER_NODE`, or a specific node. For node, use the official hostname from domain name server, and not an IP addresses or fully qualified domain name.
- `USER_ROLE` must be one of these three values: `MONITOR`, `PACKAGE_ADMIN`, or `FULL_ADMIN`.

---

### NOTE

`MONITOR` and `FULL_ADMIN` can only be set in the cluster configuration file and they apply to the entire cluster. `PACKAGE_ADMIN` can be set in the cluster or a package configuration file. If set in the cluster configuration file, `PACKAGE_ADMIN` applies to all configured packages. If set in a package configuration file, `PACKAGE_ADMIN` applies to that package only.

---

---

**NOTE**

---

You do not have to halt the cluster or package to configure or modify access control policies.

For example:

```
USER_NAME john
USER_HOST bit
USER_ROLE PACKAGE_ADMIN
```

If these policies are defined in the cluster configuration file, it grants the `PACKAGE_ADMIN` role for any package to user `john` on node `bit`. User `john` also has the `MONITOR` role for the entire cluster.

If this policy is defined in the package configuration for `PackageA`, then user `john` from node `bit` has `PACKAGE_ADMIN` role only for `PackageA`. User `john` also has the `MONITOR` role for the entire cluster.

You will not be allowed to configure any specific roles that overlap. For example, user `john` cannot be explicitly given two roles. Serviceguard will fail applying the configuration with an error if you do. It is acceptable for `ANY_USER` and `john` to be given different roles.

For example, in the cluster configuration file:

```
# Policy 1
USER_NAME john
USER_HOST bit
USER_ROLE PACKAGE_ADMIN

# Policy 2
USER_NAME john
USER_HOST bit
USER_ROLE MONITOR
```

## Preparing Your Systems

```
# Policy 3  
  
USER_NAME ANY_USER  
  
USER_HOST ANY_SERVICEGUARD_NODE  
  
USER_ROLE MONITOR
```

In the above example, the configuration will fail because user `john` is assigned two roles. Policy 2 is also redundant because `PACKAGE_ADMIN` already includes the role `MONITOR`.

Policy 3 does not conflict with either policy even though `ANY_USER` on `ANY_SERVICEGUARD_NODE` includes user `john`.

Plan the clusters roles and validate them as soon as possible. Depending on the organization's security policy, it may be easiest to create group logins. For example, you could create a `MONITOR` role for user `operator1` from `ANY_CLUSTER_NODE`. Then you could give this login name and password to everyone who will need to monitor your clusters.

Use caution when defining access to `ANY_SERVICEGUARD_NODE`. This will allow access from any node on the subnet.

## Setting up the Quorum Server

If you are using a quorum server for tie-breaking, the quorum server software has to be running during cluster configuration on a system other than the nodes on which your cluster will be running. If you are using a lock LUN for tie-breaking, skip ahead to the section entitled “Setting up the Lock LUN.”

It is recommended that the node on which the QS is running be in the same subnet as the clusters for which it is providing services. This will help prevent any network delays which could affect quorum server operation. If you use a different subnet, you may experience network delays which may cause quorum server timeouts. To prevent these timeouts, you can use the `QS_TIMEOUT_EXTENSION` parameter in the cluster ASCII file to increase the quorum server timeout interval.

If the network used to connect to the quorum server is a cluster heartbeat network, ensure that at least one other network is also a heartbeat network so that both quorum server and heartbeat communication are not likely to fail at the same time.

Perform the below listed procedures on the quorum server. Do not perform the procedures on the nodes in your clusters.

---

### NOTE

The term *quorum server* encompasses both the quorum service and the quorum system (server) where the quorum service is installed and operating.

---

## Installing the Quorum Server

Use the Linux `rpm` command to install the quorum server, product number B8467BA, on the system or systems where it will be running. You do not need to install the product on nodes that are simply using quorum services. More details on installation are found in the *Quorum Server Release Notes*.

The quorum server executable file, `qs`, is installed in the `/usr/local/qs/bin` directory for Red Hat or `/opt/qs/bin` for SuSE. When the installation is complete, you need to create an authorization file on the server where the QS will be running to allow specific host systems to obtain quorum services. The *required* pathname for this file is `/usr/local/qs/conf/qs_authfile` for Red Hat or `/opt/qs/conf/qs_authfile` for SuSE. Add to the file the names of all cluster nodes that will access cluster services from this quorum server. Enter one node per line, or enter “+” (followed by a **CR**) to allow access by all nodes. Use one line per node, as in the following example:

```
ftsys9.localdomain.com
ftsys10.localdomain.com
```

---

### NOTE

The quorum server reads the authorization file at startup. Whenever you modify the file `qs_authfile`, run the following command to force a re-read of the file. For example on a Red Hat distribution:

```
# /usr/local/qs/bin/qs -update
```

On a SuSE distribution:

```
# /opt/qs/bin/qs -update
```

---

## Running the Quorum Server

The quorum server must be running during the following cluster operations:

- when the `cmquerycl` command is issued.
- when the `cmapplyconf` command is issued.
- when there is a cluster re-formation.

By default, quorum server run-time messages go to `stdout` and `stderr`. It is suggested that you capture these messages by redirecting `stdout` and `stderr` to the file `/var/log/qs/qs.log`.

You must have root permission to execute the quorum server.

Configure the quorum server to start up any time the system on which it is installed restarts or reboots. Do this by creating an entry like the following in the `/etc/inittab` file:

For Red Hat distributions:

```
qs:345:respawn:/usr/local/qs/bin/qs >> /var/log/qs/qs.log 2>&1
```

For SuSE distributions:

```
qs:345:respawn:/opt/qs/bin/qs >> /var/log/qs/qs.log 2>&1
```

Use the following command to start the quorum server:

```
# telinit q
```

When the command is complete, the prompt appears.

Verify the quorum server is running by checking the `qs.log` file. For example on a SuSE system:

```
# cat /var/log/qs/qs.log
```

The log should contain entries like the following indicating the quorum server has started:

```
Oct 04 12:25:06:0:Starting Quorum Server
Oct 04 12:25:09:0:Server is up and waiting for connections at
port 1238
```

Refer to Chapter 3 for a complete discussion of how the quorum server operates, and see the section “Specifying a Quorum Server” under “Configuring the Cluster” later in this chapter for a description of how to use the `cmquerycl` command to specify a quorum server in the cluster ASCII file.

## Creating a Package for the Quorum Server

You can run the Quorum Server as a package in another cluster. In fact, a QS package running on one cluster can provide quorum services for any number of other clusters. You can add the Quorum Server to an existing cluster by creating a package with QS as the monitored service. Use the following procedure:

1. Install the Quorum Server software on all nodes, as described above.
2. In the configuration directory (\$SGCONF), create a subdirectory for the QS package, then change into it:

```
# mkdir qs-pkg
```

```
# cd qs-pkg
```

3. Create a package ASCII file by using the cmmakepkg command:

```
# cmmakepkg -P qs-pkg.config
```

4. Edit the file using the parameters in the following table.

Parameter	Value
PACKAGE_NAME	qs-pkg
PACKAGE_TYPE	FAILOVER
FAILOVER_POLICY	CONFIGURED_NODE
FAILBACK_POLICY	MANUAL
NODE_NAME	*
AUTO_RUN	YES
LOCAL_LAN_FAILOVER_ALLOWED	YES
NODE_FAIL_FAST_ENABLED	NO
RUN_SCRIPT	\$SGCONF/qs-pkg/qs-pkg.ct1
RUN_SCRIPT_TIMEOUT	NO_TIMEOUT
HALT_SCRIPT	\$SGCONF/qs-pkg/qs-pkg.ct1
HALT_SCRIPT_TIMEOUT	NO_TIMEOUT
SERVICE_NAME	qs



Parameter	Value
SERVICE_FAIL_FAST_ENABLED	NO
SERVICE_HALT_TIMEOUT	10
SUBNET	Specify your subnet here.

5. Create a control script in the same directory:

```
# cmmakepkg -s qs-pkgctl
```

6. Edit the file using the parameters in the following table.

Parameter	Value
IP[0]	IP address to be used when accessing the Quorum Server
SUBNET[0]	Specify your subnet here
SERVICE_NAME[0]	"qs"
SERVICE_CMD[0]	HP-UX: "/usr/sbin/qs >> /var/adm/qs/qs.log 2>&1" Red Hat: "/usr/local/qs/bin/qs >> /var/log/qs/qs.log 2>&1" SuSE: "/opt/qs/bin/qs >> /var/log/qs/qs.log 2>&1"
SERVICE_RESTART	"-R"

7. Run the cluster and start the Quorum Server package.

---

## Setting up the Lock LUN

The lock LUN requires a partition of one cylinder of at least 100K defined (via the `fdisk` command) as type Linux (83). You will need the pathnames for the lock LUN as it is seen on each cluster node. On one node, use the `fdisk` command to define a partition of 1 cylinder, type 83, on this LUN. Here is an example:

Respond to the prompts as shown in the following table to set up the lock LUN partition:

```
# fdisk <Lock LUN Device File>
```

**Table 5-1**                      **Changing Linux Partition Types**

	Prompt	Response	Action Performed
1.	Command (m for help):	<b>n</b>	Create new partition
2.	Partition number (1-4):	<b>1</b>	Partition affected
3.	Hex code (L to list codes):	<b>83</b>	Set partition to type to Linux, default
	Command (m for help):	<b>1</b>	Define first partition
	Command (m for help):	<b>1</b>	Set size to 1 cylinder
	Command (m for help):	<b>p</b>	Display partition data
	Command (m for help):	<b>w</b>	Write data to the partition table

The following example of the `fdisk` dialog shows that the disk on the device file `/dev/sdc` is set to Smart Array type partition, and appears as follows:

```
# fdisk /dev/sdc

Command (m for help): n
Partition number (1-4): 1
HEX code (type L to list codes): 83
Command (m for help): 1
Command (m for help): 1
```

```
Command (m for help): p
Disk /dev/sdc: 64 heads, 32 sectors, 4067 cylinders
Units = cylinders of 2048 * 512 bytes

Device Boot      Start          End      Blocks   Id  System
/dev/sdc1         1            1       1008    83   Linux

Command (m for help): w
The partition table has been altered!
```

---

## NOTE

- Do not try to use LVM to configure the lock LUN.
- The partition type must be 83.
- Do not create any filesystem on the partition used for the lock LUN.
- Do not use `md` to configure multiple paths to the lock LUN.

---

To transfer the disk partition format to other nodes in the cluster use the command:

```
# sfdisk -R <device>
```

where `<device>` corresponds to the same physical device as on the first node. For example, if `/dev/sdc` is the device name on the other nodes use the command:

```
# sfdisk -R /dev/sdc
```

You can check the partition table by using the command:

```
# fdisk -l /dev/sdc
```

---

## NOTE

`fdisk` may not be available for SUSE on all platforms. In this case, using `YAST2` to set up the partitions is acceptable.

---

## Implementing Channel Bonding (Red Hat)

Use the procedures included in this section to implement channel bonding on Red Hat installations. If you are using a SuSE distribution, skip ahead to the next section.

Channel bonding of LAN interfaces is implemented by the use of the bonding driver, which is installed in the kernel at boot time. With this driver installed, the networking software recognizes bonding definitions that are created in the `/etc/sysconfig/network-scripts` directory for each bond. For example, the file named `ifcfg-bond0` defines `bond0` as the master bonding unit, and the `ifcfg-eth0` and `ifcfg-eth1` scripts define each individual interface as a slave.

Bonding can be defined in different modes. Mode 0, which is used for load balancing, uses all slave devices within the bond in parallel for data transmission. This can be done when the LAN interface cards are connected to an Ethernet switch such as the HP ProCurve switch, with the ports on the switch configured as Fast EtherChannel trunks. Two switches should be cabled together as an HA grouping to allow package failover.

For high availability, in which one slave serves as a standby for the bond and the other slave transmits data, install the bonding module in mode 1. This is most appropriate for dedicated heartbeat connections that are cabled through redundant network hubs or switches that are cabled together.

For more networking information on bonding, see:

RedHat 3:

`/usr/src/linux-2.4/Documentation/networking/bonding.txt`

RedHat 4:

`/usr/share/doc/kernel-2.6.9/Documentation/networking/bonding.txt`

---

### NOTE

HP recommends that you do the bonding configuration from the system console, because you will need to restart networking from the console when the configuration is done.

---

## Sample Configuration

Configure the following files to support LAN redundancy. For a single failover only one bond is needed.

1. Create a `bond0` file, `ifcfg-bond0`.

Create the configuration in the `/etc/sysconfig/network-scripts` directory. For example, in the file, `ifcfg-bond0`, `bond0` is defined as the master (for your installation, substitute the appropriate values for your network instead of `192.168.1`).

Include the following information in the `ifcfg-bond0` file:

```
DEVICE=bond0
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

2. Create an `ifcfg-ethn` file for each interface in the bond. All interfaces should have `SLAVE` and `MASTER` definitions. For example, in a bond that uses `eth0` and `eth1`, edit the `ifcfg-eth0` file to appear as follows:

```
DEVICE=eth0
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

Edit the `ifcfg-eth1` file to appear as follows:

```
DEVICE=eth1
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

3. Add the following lines to `/etc/modules.conf`:

```
alias bond0 bonding
options bond0 miimon=100 mode=1
```

Use `MASTER=bond1` for `bond1` if you have configured a second bonding interface, then add the following after the first bond (`bond0`):

```
options bond1 -o bonding1 miimon=100 mode=1
```

---

**NOTE**

During configuration, you need to make sure that the active slaves for the same bond on each node are connected the same hub or switch. You can check on this by examining the file `/proc/net/bondx/info` on each node. This file will show the active slave for bond `x`.

---

## Restarting Networking

Restart the networking subsystem. *From the console* of either node in the cluster, execute the following command on a Red Hat system:

```
# /etc/rc.d/init.d/network restart
```

---

**NOTE**

It is better not to restart the network from outside the cluster subnet, as there is a chance the network could go down before the command can complete.

---

The command prints `bringing up network` statements.

If there was an error in any of the bonding configuration files, the network might not function properly. If this occurs, check each configuration file for errors, then try to restart the network again.

## Viewing the Configuration

You can test the configuration and transmit policy with `ifconfig`. For the example, execution on the above created configuration, the display should appear like this:

```
# /sbin/ifconfig
```

```
bond0      Link encap:Ethernet  HWaddr 00:C0:F0:1F:37:B4
inet addr:192.168.1.1 Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
RX packets:7224794 errors:0 dropped:0 overruns:0 frame:0
TX packets:3286647 errors:1 dropped:0 overruns:1 carrier:0
collisions:0 txqueuelen:0
```

```
eth0       Link encap:Ethernet  HWaddr 00:C0:F0:1F:37:B4
inet addr:192.168.1.1 Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:3573025 errors:0 dropped:0 overruns:0 frame:0
TX packets:1643167 errors:1 dropped:0 overruns:1 carrier:0
collisions:0 txqueuelen:100
Interrupt:10 Base address:0x1080
```

```
eth1       Link encap:Ethernet  HWaddr 00:C0:F0:1F:37:B4
inet addr:192.168.1.1 Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:3651769 errors:0 dropped:0 overruns:0 frame:0
TX packets:1643480 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
Interrupt:9 Base address:0x1400
```

## Implementing Channel Bonding (SLES9 and SLES10)

If you are using a Red Hat distribution, use the procedures described in the previous section. The following applies only to the SLES9 and SLES10 distributions.

First run `yast/yast2` and configure ethernet devices as DHCP so they create the `ifcfg-eth-id-<mac>` files.

Next modify each of `ifcfg-eth-id-<mac>` files that you want to bond, they are located in `/etc/sysconfig/network`, and change them from:

```
BOOTPROTO='dhcp'
MTU=' '
REMOTE_IPADDR=' '
STARTMODE='onboot'
UNIQUE='gZD2.ZqnB7JKTdX0'
_nm_name='bus-pci-0000:00:0b.0'
```

To:

```
BOOTPROTO='none'
STARTMODE='onboot'
UNIQUE='gZD2.ZqnB7JKTdX0'
_nm_name='bus-pci-0000:00:0b.0'
```

---

### NOTE

Be sure to leave the `UNIQUE` and the `_nm_name` alone. You can also leave in the `MTU` and `REMOTE_IPADDR` as long as they are not set.

---

Next in `/etc/sysconfig/network` edit your `ifcfg-bond0` file so it looks like the following:

```
BROADCAST='172.16.0.255'
BOOTPROTO='static'
IPADDR='172.16.0.1'
```



```
MTU=' '  
NETMASK='255.255.255.0'  
NETWORK='172.16.0.0'  
REMOTE_IPADDR=' '  
STARTMODE='onboot'  
BONDING_MASTER='yes'  
BONDING_MODULE_OPTS='miimon=100 mode=1'  
BONDING_SLAVE0='eth0'  
BONDING_SLAVE1='eth1'
```

The above example configures bond0 with mii monitor equal to 100 and mode active-backup. Adjust the IP, BROADCAST, NETMASK, NETWORK accordingly for your configuration. The new config options as you can see are BONDING\_MASTER, BONDING\_MODULE\_OPTS, BONDING\_SLAVE. The BONDING\_MODULE\_OPTS is the additional options you want to pass to the bonding module. You can not pass max\_bonds as an option. This is not needed as the ifup script will load the module for each bond needed.

Next is the BONDING\_SLAVE. This tells ifup which ethernet devices to enslave to bond0. So if you wanted to bond four ethernet devices you would add:

```
BONDING_SLAVE2='eth2 '  
BONDING_SLAVE3='eth3 '
```

---

**NOTE**

Use ifconfig to find the relationship of eth IDs and the MAC address.

---

For more networking information on bonding, see

```
/usr/src/linux<kernel_version>/Documentation/networking/bonding.  
txt
```

## Restarting Networking

Restart the networking subsystem. *From the console* of either node in the cluster, execute the following command on a SUSE system:

```
# /etc/init.d/network restart
```

---

**NOTE**

It is better not to restart the network from outside the cluster subnet, as there is a chance the network could go down before the command can complete.

---

If there was an error in any of the bonding configuration files, the network might not function properly. If this occurs, check each configuration file for errors, then try to restart the network again.

## Creating the Logical Volume Infrastructure

Serviceguard makes use of shared disk storage. This is set up to provide high availability by using redundant data storage and redundant paths to the shared devices. Storage for a Serviceguard package is logically composed of LVM Volume Groups that are activated on a node as part of starting a package on that node. Storage is generally configured on logical units (LUNs).

Disk storage for Serviceguard packages is built on shared disks that are cabled to multiple cluster nodes. These are separate from the private Linux root disks, which include the boot partition and root file systems. To provide space for application data on shared disks, create disk partitions using the `fdisk`, and build logical volumes with LVM.

You can build a cluster (next section) before or after defining volume groups for shared data storage. If you create the cluster first, information about storage can be added to the cluster and package configuration files after the volume groups are created.

See “Volume Managers for Data Storage” on page 75 for an overview of volume management in HP Serviceguard for Linux. The sections that follow explain how to do the following tasks:

- “Displaying Disk Information” on page 149
- “Creating Partitions” on page 149
- “Enabling VG Activation Protection” on page 152
- “Building Volume Groups: Example for Smart Array Cluster Storage (MSA 500 Series)” on page 153
- “Building Volume Groups and Logical Volumes” on page 156
- “Distributing the Shared Configuration to all Nodes” on page 157
- “Testing the Shared Configuration” on page 158
- “Storing Volume Group Configuration Data” on page 160
- “Setting up Disk Monitoring” on page 161

---

**CAUTION**

The minor numbers used by the LVM volume groups must be the same on all cluster nodes. This means that if there are any non-shared volume groups in the cluster, create the same number of them on all nodes, and create them *before* you define the shared storage.

---

---

**NOTE**

Except as noted in the sections that follow, you perform the LVM configuration of shared storage on only one node. The disk partitions will be visible on other nodes as soon as you reboot those nodes. After you've distributed the LVM configuration to all the cluster nodes, you will be able to use LVM commands to switch volume groups between nodes. (To avoid data corruption, a given volume group must be active on only one node at a time).

For multipath information, see the “Multipath for Storage” section of the latest version of the *Serviceguard for Linux A.11.16 Release Notes*.

---

## Displaying Disk Information

To display a list of configured disks, use the following command:

```
# fdisk -l
```

You will see output such as the following:

```
Disk /dev/sda: 64 heads, 32 sectors, 8678 cylinders
Units = cylinders of 2048 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	1001	1025008	83	Linux
/dev/sda2		1002	8678	7861248	5	Extended
/dev/sda5		1002	4002	3073008	83	Linux
/dev/sda6		4003	5003	1025008	82	Linux swap
/dev/sda7		5004	8678	3763184	83	Linux

```
Disk /dev/sdb: 64 heads, 32 sectors, 8678 cylinders
Units = cylinders of 2048 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
--------	------	-------	-----	--------	----	--------

```
Disk /dev/sdc: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

```
Disk /dev/sdd: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

In this example, the disk described by device file `/dev/sda` has already been partitioned for Linux, into partitions named `/dev/sda1` - `/dev/sda7`. The second internal device `/dev/sdb` and the two external devices `/dev/sdc` and `/dev/sdd` have not been partitioned.

---

### NOTE

`fdisk` may not be available for SUSE on all platforms. In this case, using YAST2 to set up the partitions is acceptable.

---

## Creating Partitions

You must define a partition on each disk device (individual disk or LUN in an array) that you want to use for your shared storage. Use the `fdisk` command for this.

The following steps create the new partition:

1. Run `fdisk`, specifying your device file name in place of `<DeviceName>`:

```
# fdisk <DeviceName>
```

Respond to the prompts as shown in the following table, to define a partition:

	Prompt	Response	Action Performed
1.	Command (m for help):	<b>n</b>	Create a new partition
2.	Command action e extended p primary partition (1-4)	<b>p</b>	Creation a primary partition
3.	Partition number (1-4):	<b>1</b>	Create partition 1
4.	First cylinder (1- <i>nn</i> , default 1):	<b>Enter</b>	Accept the default starting cylinder 1
5.	Last cylinder or +size or +sizeM or +sizeK (1- <i>nn</i> , default <i>nn</i> ):	<b>Enter</b>	Accept the default, which is the last cylinder number
	Command (m for help):	<b>p</b>	Display partition data
	Command (m for help):	<b>w</b>	Write data to the partition table

The following example of the `fdisk` dialog shows that the disk on the device file `/dev/sdc` is configured as one partition, and appears as follows:

```
# fdisk /dev/sdc1
```

```
Command (m for help): n
```

```
Command action
```

```
  e  extended
```

```
  p  primary partition (1-4) p
```

```
Partition number (1-4): 1
```

```
First cylinder (1-4067, default 1): Enter
```

```
Using default value 1
```

```
Last cylinder or +size or +sizeM or +sizeK (1-4067, default 4067): Enter
```

```
Using default value 4067
```

```
Command (m for help): p
```

```
Disk /dev/sdc: 64 heads, 32 sectors, 4067 cylinders
Units = cylinders of 2048 * 512 bytes
```

```
Device Boot      Start          End      Blocks   Id  System
/dev/sdc1                1         4067     4164592    83  Linux
```

```
Command (m for help): w
The partition table has been altered!
```

2. Respond to the prompts as shown in the following table to set a partition type:

	Prompt	Response	Action Performed
1.	Command (m for help):	t	Set the partition type
2.	Partition number (1-4):	1	Partition affected
3.	Hex code (L to list codes):	8e	Set partition to type to Linux LVM
	Command (m for help):	p	Display partition data
	Command (m for help):	w	Write data to the partition table

The following example of the `fdisk` dialog describes that the disk on the device file `/dev/sdc` is set to Smart Array type partition, and appears as follows:

```
# fdisk /dev/sdc1
```

```
Command (m for help): t
Partition number (1-4): 1
HEX code (type L to list codes): 8e
```

```
Command (m for help): p
Disk /dev/sdc: 64 heads, 32 sectors, 4067 cylinders
Units = cylinders of 2048 * 512 bytes
```

```
Device Boot      Start          End      Blocks   Id  System
/dev/sdc1                1         4067     4164592    8e  Linux LVM
```

```
Command (m for help): w
The partition table has been altered!
```

3. Repeat this process for each device file that you will use for shared storage.

```
# fdisk /dev/sdd
# fdisk /dev/sdf
# fdisk /dev/sdg
```

4. If you will be creating volume groups for internal storage, make sure to create those partitions as well, and create those volume groups before you define the shared storage.

```
# fdisk /dev/sddb
```

---

**NOTE**

`fdisk` may not be available for SUSE on all platforms. In this case, using YAST2 to set up the partitions is acceptable.

---

## Enabling VG Activation Protection

Follow these steps to enable activation protection for volume groups on RedHat 4, SUSE SLES 9, and SUSE SLES 10 systems:

---

**NOTE**

Perform these steps on *each* node.

---

- Step 1.** Edit `/etc/lvm/lvm.conf` and add the following line:

```
tags { hosttags = 1 }
```

- Step 2.** Create the file `/etc/lvm/lvm_$(uname -n).conf`

- Step 3.** Add the following line to the file you created in step 2:

```
activation { volume_list=["@node"] }
```

where *node* is the value of `uname -n`.

- Step 4.** Run `vgscan`:

```
# vgscan
```



---

**NOTE**

At this point, the setup for VG activation protection is complete. As of Serviceguard for Linux A.11.16.07, the package control script adds a tag matching the value of *node* (as specified in step 3 above) when it activates the volume group, and deletes the tag when it deactivates it, preventing the volume group from being activated by more than one node at the same time. The purpose of the steps that follow is to verify that the setup has been done correctly.

---

- Step 5.** Add the tag and verify that it's valid, and that the volume group activates as expected, for example:

```
# vgchange --addtag $(uname -n) vgpkgA  
  
# vgs -o +tags vgpkgA  
  
# vgchange -a y vgpkgA
```

The VG `Tags` column in the output should contain the value of `uname -n`.

- Step 6.** Deactivate the volume group and delete the tag:

```
# vgchange -a n vgpkgA  
  
# vgchange --deltag $(uname -n) vgpkgA
```

## Building Volume Groups: Example for Smart Array Cluster Storage (MSA 500 Series)

Use Logical Volume Manager (LVM) on your system to create volume groups that can be activated by Serviceguard packages. This section provides an example of creating Volume Groups on LUNs created on MSA 500 Series storage. For more information on LVM, see to the *Logical Volume Manager How To* at [www.linuxdoc.org/HowTo/LMV-HOWTO.html](http://www.linuxdoc.org/HowTo/LMV-HOWTO.html)

Before you start, partition your LUNs and label them with a partition type of `8e` (Linux LVM). Use the type `t` parameter of the `fdisk` command to change from the default of `83` (Linux).

Do the following on one node:

1. Update the LVM configuration and create the `/etc/lvmtab` file. You can omit this step if you have previously created volume groups on this node.

```
# vgscan
```

---

**NOTE**

---

The files `/etc/lvmtab` and `/etc/lvmtab.d` may not exist on some distributions. In that case, ignore references to these files.

2. Create LVM **physical volumes** on each LUN. For example:

```
# pvcreate -f /dev/cciss/c0d1p1
```

```
# pvcreate -f /dev/cciss/c0d2p1
```

```
# pvcreate -f /dev/cciss/c0d3p1
```

3. Check whether there are already volume groups defined on this node. Be sure to give each volume group a unique name.

```
# vgdisplay
```

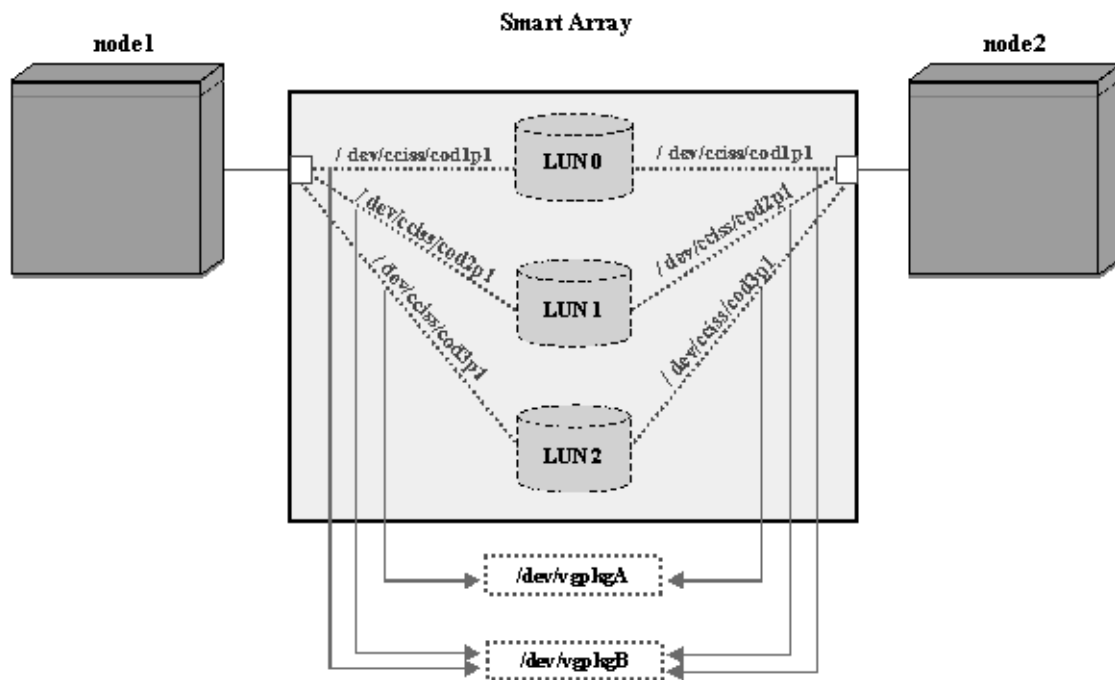
4. Create separate volume groups for each Serviceguard package you will define. In the following example, the LUNs `/dev/cciss/c0d1p1` and `/dev/cciss/c0d2p1`, which are defined earlier as physical volumes, are added to volume group `vgpkgA` and `/dev/cciss/c0d3p1` is added to `vgpkgB`:

```
# vgcreate /dev/vgpkgA /dev/cciss/c0d1p1 /dev/cciss/c0d2p1
```

```
# vgcreate /dev/vgpkgB /dev/cciss/c0d3p1
```

Figure 5-1 shows these two volume groups as they are constructed for the MSA500 Storage.

**Figure 5-1 MSA500 Storage Volume Groups**



## Building Volume Groups and Logical Volumes

- Step 1.** Use Logical Volume Manager (LVM) to create volume groups that can be activated by Serviceguard packages.

For an example showing volume-group creation on LUNs, see “Building Volume Groups: Example for Smart Array Cluster Storage (MSA 500 Series)” on page 153. (For Fibre Channel storage you would use device-file names such as those used in the section “Creating Partitions” on page 149.) See also the *Logical Volume Manager How To* at [www.linuxdoc.org/HowTo/LMV-HOWTO.html](http://www.linuxdoc.org/HowTo/LMV-HOWTO.html)

- Step 2.** On Linux distributions that support it, enable activation protection for volume groups. See “Enabling VG Activation Protection” on page 152.

- Step 3.** To store data on these volume groups you must create logical volumes. The following creates a 500 Megabyte logical volume named `/dev/vgpkgA/lvol1` and a one Gigabyte logical volume named `/dev/vgpkgA/lvol2` in volume group `vgpkgA`:

```
# lvcreate -L 500M vgpkgA
# lvcreate -L 1G vgpkgA
```

- Step 4.** Create a file system on one of these logical volumes, and mount it in a newly created directory:

```
# mke2fs -j /dev/vgpkgA/lvol1
# mkdir /extra
# mount -t ext3 /dev/vgpkgA/lvol1 /extra
```

---

### NOTE

Serviceguard for Linux supports the **ext3**, **reiserfs** and **XFS** journaling filesystems. Reiserfs and XFS may not be available on Red Hat. All three are available on SLES9.

---

- Step 5.** To test that the file system `/extra` was created correctly and with high availability, you can create a file on it, and read it.

```
# echo "Test of LVM" >> /extra/LVM-test.ascii
# cat /extra/LVM-test.ascii
```

---

**NOTE**

Be careful if you use YAST or YAST2 to configure volume groups, as that may cause *all* volume groups on that system to be activated. After running YAST or YAST2, check to make sure that volume groups for Serviceguard packages not currently running have not been activated, and use LVM commands to deactivate any that have. For example, use the command `vgchange -a n /dev/sgvg00` to deactivate the volume group `sgvg00`.

---

## Distributing the Shared Configuration to all Nodes

The goal in setting up a logical volume infrastructure is to build a set of volume groups that can be activated on multiple nodes in the cluster. To do this, you need to build the same LVM volume groups on any nodes that will be running the same package.

---

**NOTE**

The minor numbers used by the LVM volume groups must be the same on all cluster nodes. They will if all the nodes have the same number of unshared volume groups.

---

To distribute the shared configuration, follow these steps:

1. Unmount and deactivate the volume group, and remove the tag if necessary. For example, to deactivate only `vgpkgA`:

```
# umount /extra
# vgchange -a n vgpkgA
# vgchange --deltag $(uname -n) vgpkgA
```

---

**NOTE**

Use `vgchange --addtag` and `vgchange --deltag` commands only if you are implementing VG activation protection. Remember that VG activation protection, if used, must be implemented on *each* node.

---

2. To get the node `ftsys10` to see the new disk partitioning that was done on `ftsys9`, reboot:

```
# reboot
```

The partition table on the rebooted node is then rebuilt using the information placed on the disks when they were partitioned on the other node.

---

**NOTE**

---

You *must* reboot at this time.

3. Run `vgscan` to make the LVM configuration visible on the new node and to create the LVM database on `/etc/lvmtab` and `/etc/lvmtab.d`. For example, on `ftsys10`:

```
# vgscan
```

## Testing the Shared Configuration

When you have finished the shared volume group configuration, you can test that the storage is correctly sharable as follows:

- Step 1.** On `ftsys9`, activate the volume group, mount the file system that was built on it, write a file in the shared file system and look at the result:

```
# vgchange --addtag $(uname -n) vgpkgB
# vgchange -a y vgpkgB
# mount /dev/vgpkgB/lvol1 /extra
# echo 'Written by' `hostname` `on` `date` > /extra/datestamp
# cat /extra/datestamp
```

You should see something like the following, showing the datestamp written by the other node:

```
Written by ftsys9.mydomain on Mon Jan 22 14:23:44 PST 2006
```

Now unmount the volume group again:

```
# umount /extra
# vgchange -a n vgpkgB
# vgchange --deltag $(uname -n) vgpkgB
```

- Step 2.** On `ftsys10`, activate the volume group, mount the file system, write a datestamp on to the shared file, and then look at the content of the file:

```
# vgchange --addtag $(uname -n) vgpkgB

# vgchange -a y vgpkgB
# mount /dev/vgpkgB/lvol1 /extra
# echo 'Written by' 'hostname' 'on' 'date' >>
  /extra/datestamp
# cat /extra/datestamp
```

You should see something like the following, including the datestamp written by the other node:

```
Written by ftsys9.mydomain on Mon Jan 22 14:23:44 PST 2006
```

```
Written by ftsys10.mydomain on Mon Jan 22 14:25:27 PST 2006
```

Now unmount the volume group again, and remove the tag you added in step 1:

```
# umount /extra

# vgchange -a n vgpkgB

# vgchange --deltag $(uname -n) vgpkgB
```

---

## NOTE

The VG activation protection feature of Serviceguard for Linux requires that you add the tag as shown at the beginning of the above steps when you manually activate a volume group. Similarly, you must remove the tag when you deactivate a volume group that will be used in a package (as shown at the end of each step). As of Serviceguard for Linux A.11.16.07, a tag matching the `uname -n` value of the owning node is automatically added to each volume group defined in a package control script when the package runs; the tag is deleted when the package is halted. The command `vgs -o +tags vgname` will display any tags that are set for a volume group.

---

## Storing Volume Group Configuration Data

When you create volume groups, LVM creates a backup copy of the volume group configuration on the configuration node. In addition, you should create a backup of configuration data on all other nodes where the volume group might be activated by using the `vgcfgbackup` command:

```
# vgcfgbackup vgpkgA vgpkgB
```

If a disk in a volume group must be replaced, you can then restore the disk's meta data by using the `vgcfgrestore` command. The procedure is described under “Replacing Disks” in the “Troubleshooting” chapter.

Also please to be sure to follow the instructions in the “Preventing Boot-Time VG Scan” section of the Serviceguard for Linux Release Notes.



## Setting up Disk Monitoring

HP Serviceguard for Linux includes a Disk Monitor which you can use to detect problems in disk connectivity. This lets you fail a package from one node to another in the event of a disk link failure. To identify the set of disks you wish to monitor, you run a script after all packages are configured. (The script is run again any time you add, change or remove packages.)

The process for configuring disk monitoring is described in “Creating a Disk Monitor Configuration” on page 206.

For information on configuring lock LUNs, see “Setting up the Lock LUN” on page 138.

## Configuring the Cluster

Use the `cmquerycl` command to specify a set of nodes to be included in the cluster and to generate a template for the cluster configuration file. The following is an example of the command as issued from node `ftsys9`:

```
# cmquerycl -v -C $SGCONF/clust1.config \  
-n ftsys9 -n ftsys10 -q lp-qs
```

The example creates an ASCII template file in the default cluster configuration directory `$SGCONF`. The ASCII file is partially filled in with the names and characteristics of cluster components on the two nodes `ftsys9` and `ftsys10`. The use of the `-q` option lets you specify the quorum server host name, which will be included as the value of the `QS_HOST` parameter in the ASCII file.

Edit the filled-in cluster characteristics as needed to define the desired cluster. It is strongly recommended that you edit the file to send heartbeat over all possible networks, as shown in the following example.

---

### NOTE

In a larger or more complex configuration with many nodes and networks connected to the cluster, the `cmquerycl` command may require several minutes to complete. In order to speed up the configuration process, you can direct the command to return selected information only by using the `-w` option:

`-w local` lets you specify local network probing, in which LAN connectivity is verified between interfaces within each node only.

`-w full` lets you specify full network probing, in which actual connectivity is verified among all LAN interfaces on all nodes in the cluster. For complete details, refer to the man page on `cmquerycl`.

---

## Using Serviceguard Manager to Configure the Cluster

### Before you start:

1. Configure the volume groups.
2. If you are using a quorum server to manage the cluster lock, make sure it is running.

To configure the cluster using Serviceguard Manager, proceed as follows. Online Help is available at each step to help you make decisions.

1. Create a session on Serviceguard Manager. Select the option for discovering unused nodes. On the map or tree, from the list of unused nodes, select the one where you want to start the cluster.
2. From the Actions menu, choose Configuring.

After you give the node's root password, the Configuration screen will open, and you will be guided through the process. Each tab contains related information. Serviceguard Manager discovers much of the information, so you can choose from available options, such as lists of volume groups, networks, and nodes.

3. When you complete your information, click Apply. If there are errors, they are displayed in a log window. If not, the log displays a "successful" message, and the binary configuration is automatically distributed to the nodes.
4. After a Refresh, the new cluster configuration and status information appears in the tree, map and Properties.

To modify or delete the configuration, select the cluster on the tree or map, and choose Configuring from the Actions menu.

## Specifying a Quorum Server

If you will be using a quorum server, be sure to specify the `-q qshost` option with the `cmquerycl` command. For example on a Red Hat distribution:

```
# cmquerycl -v -n lp001 -n lp002 -q lp-qs \  
-C $SGCONF/lpcluster.config
```

## Specifying a Lock LUN

If you will be using a lock LUN, be sure to specify the `-L lock_lun_device` option with the `cmquerycl` command. If the name of the device is the same on all nodes, enter the option before the node names, as in the following example:

```
# cmquerycl -v -L /dev/cciss/c0d0p1 -n lp01 -n lp02 \
-C $SGCONF/lpcluster.config
```

If the name of the device is different on the different nodes, use the option to specify each device file following each node name, as in the following example:

```
# cmquerycl -v -n lp01 -L /dev/cciss/c0d0p1 \
-n lp02 -L /dev/cciss/c0d0p2 \
-C $SGCONF/lpcluster.config
```

## Cluster Configuration Template File

When you install Serviceguard for the first time on a node, you do not have a cluster. Without cluster configuration, however, you cannot have an Access Control Policy.

You can, however, create a `cmclnodelist` file to act as a “bootstrap” monitor access. Bootstrap files are useful if you are doing a rolling upgrade, so the nodes with older versions can still access the newer cluster nodes. Monitor access in a bootstrap file also means the node can appear in Serviceguard Manager; you can see information about it in Properties, and you can configure it into a cluster after you give the root password.

### To create a bootstrap file:

The `cmclnodelist` file is not created by default in new installations. If you create this bootstrap file you should add a comment such as the following:

```
#####
# Do Not Edit This File
# This is only a temporary file to bootstrap an unconfigured
# node with Serviceguard version A.11.16
# Once a cluster is created, Serviceguard will not consult
```

# this file.

#####

The format for entries in the `cmclnodelist` file is as follows:

```
[hostname or ip address] [user] [#Comment]
```

For example:

```
gryf root # Cluster 1, Node 1
gryf user1 # Cluster 1, Node 1
sly root # Cluster 1, Node 2
sly user1 # Cluster 1, Node 2
bit root # Administration /COM Server
```

In this example, `root` on the nodes `gryf`, `sly`, and `bit` all have root access to the node with this file. The non-root user “`user1`” has the Monitor role from nodes `gryf` and `sly`.

Serviceguard also accepts the use of a “+” in the `cmclnodelist` file which indicates that any root user on any node may configure this node and any non-root user has the Monitor role.

The following is an example of an ASCII configuration file generated with the `cmquerycl` command using the `-w full` option:

```
# *****
# ***** HIGH AVAILABILITY CLUSTER CONFIGURATION FILE *****
# ***** For complete details about cluster parameters and how to *****
# ***** set them, consult the Serviceguard manual. *****
# *****

# Enter a name for this cluster. This name will be used to identify the cluster
when viewing or manipulating it.

CLUSTER_NAME          lxcluster

# Cluster Lock Parameters
# The cluster lock is used as a tie-breaker for situations
# in which a running cluster fails, and then two equal-sized
# sub-clusters are both trying to form a new cluster. The
# cluster lock may be configured using only one of the
# following alternatives on a cluster:
#         the LUN lock disk
#         the quorum server
#
#
# Consider the following when configuring a cluster.
# For a two-node cluster, you must use a cluster lock. For
# a cluster of three or four nodes, a cluster lock is strongly
# recommended. For a cluster of more than four nodes, a
# cluster lock is recommended. If you decide to configure
# a lock for a cluster of more than four nodes, it must be
# a quorum server.
```

## Building an HA Cluster Configuration

### Configuring the Cluster

```
# LUN Lock Disk Parameters. Use the CLUSTER_LOCK_LUN parameter to define
# the device on a per node basis. The device may only be used for the purpose
# and by only a single cluster.
#
# Example for a smart array cluster disk
# CLUSTER_LOCK_LUN /dev/cciss/c0d0p1
# Example for a non smart array cluster disk
# CLUSTER_LOCK_LUN /dev/sdal

# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.
# The QS_HOST is the host name or IP address of the system
# that is running the quorum server process. The
# QS_POLLING_INTERVAL (microseconds) is the interval at which
# Serviceguard checks to make sure the quorum server is running.
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase
# the time interval after which the quorum server is marked DOWN.
#
# The default quorum server timeout is calculated from the
# Serviceguard cluster parameters, including NODE_TIMEOUT and
# HEARTBEAT_INTERVAL. If you are experiencing quorum server
# timeouts, you can adjust these parameters, or you can include
# the QS_TIMEOUT_EXTENSION parameter.
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000

# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.

NODE_NAME                lpctest3
NETWORK_INTERFACE        eth0
HEARTBEAT_IP             15.13.172.231

# Warning: There are no standby network interfaces for eth0.

NODE_NAME                lpctest4
NETWORK_INTERFACE        eth0
HEARTBEAT_IP             15.13.172.232

# Warning: There are no standby network interfaces for eth0.

# Cluster Timing Parameters (microseconds).

# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
# This default setting yields the fastest cluster reformatations.
# However, the use of the default value increases the potential
# for spurious reformatations due to momentary system hangs or
# network load spikes.
# For a significant portion of installations, a setting of
# 5000000 to 8000000 (5 to 8 seconds) is more appropriate.
# The maximum value recommended for NODE_TIMEOUT is 30000000
# (30 seconds).

HEARTBEAT_INTERVAL       1000000
NODE_TIMEOUT              2000000
```

```
# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT          600000000
NETWORK_POLLING_INTERVAL    2000000

# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the cluster.
# You can not add packages beyond this limit.
# This parameter is required.
MAX_CONFIGURED_PACKAGES     3

# Access Control Policy Parameters.
#
# Three entries set the access control policy for the cluster:
# First line must be USER_NAME, second USER_HOST, and third
# USER_ROLE.
# Enter a value after each.
#
# 1. USER_NAME can either be ANY_USER, or a maximum of
#    8 login names from the /etc/passwd file on user host.

# 2. USER_HOST is where the user can issue Serviceguard
#    commands.
#    If using Serviceguard Manager, it is the COM server.
#    Choose one of these three values: ANY_SERVICEGUARD_NODE,
#    or
#    (any) CLUSTER_MEMBER_NODE, or a specific node. For node,
#    use the official hostname from domain name server, and not
#    an IP addresses or fully qualified name.
# 3. USER_ROLE must be one of these three values:
#    * MONITOR: read-only capabilities for the cluster and
#    packages
#    * PACKAGE_ADMIN: MONITOR, plus administrative commands for
#    packages
#    in the cluster
#    * FULL_ADMIN: MONITOR and PACKAGE_ADMIN plus the
#    administrative
#    commands for the cluster.
#
# Access control policy does not set a role for configuration
# capability. To configure, a user must log on to one of the
# cluster's nodes as root (UID=0). Access control
# policy cannot limit root users' access.
#
# MONITOR and FULL_ADMIN can only be set in the cluster
# configuration file,
# and they apply to the entire cluster. PACKAGE_ADMIN can be
# set in the
# cluster or a package configuration file. If set in the
# cluster
# configuration file, PACKAGE_ADMIN applies to all configured
# packages.
```

```

# If set in a package configuration file, PACKAGE_ADMIN applies
to that
# package only.
#
# Conflicting or redundant policies will cause an error while
applying
# the configuration, and stop the process. The maximum number
of access
# policies that can be configured in the cluster is 200.
#
#
# Example: to configure a role for user john from node noir to
# administer a cluster and all its packages, enter:
# USER_NAME    john
# USER_HOST    noir
# USER_ROLE    FULL_ADMIN

USER_NAME      root
USER_HOST      ANY_SERVICEGUARD_NODE
USER_ROLE      full_admin

```

The man page for the `cmquerycl` command lists the definitions of all the parameters that appear in this file. Many are also described in the “Planning” chapter. Modify your `$SGCONF/clust1.config` file to your requirements, using the data on the cluster configuration worksheet.

In the file, keywords are separated from definitions by white space. Comments are permitted, and must be preceded by a pound sign (#) in the far left column. See the man page for the `cmquerycl` command for more details.

## Specifying Maximum Number of Configured Packages

Serviceguard preallocates memory and threads at cluster startup time. It calculates these values based on the number of packages specified in the `MAX_CONFIGURED_PACKAGES` parameter in the cluster configuration file. This value must be equal to or greater than the number of packages currently configured in the cluster. The default is 0, which means that you must enter a value if you wish to use packages. The absolute maximum number of packages per cluster is 150. Serviceguard reserves 6MB of lockable memory when you configure packages plus approximately 80KB for each package.



## Modifying Cluster Timing Parameters

The `cmquerycl` command supplies default cluster timing parameters for `HEARTBEAT_INTERVAL` and `NODE_TIMEOUT`. Changing these parameters will directly impact the cluster's reformation and failover times. It is useful to modify these parameters if the cluster is reforming occasionally due to heavy system load or heavy network traffic.

The default value of 2 seconds for `NODE_TIMEOUT` leads to a best case failover time of 30 seconds. If `NODE_TIMEOUT` is changed to 10 seconds, which means that the cluster manager waits 5 times longer to timeout a node, the failover time is increased by 5, to approximately 150 seconds. `NODE_TIMEOUT` must be at least  $2 * \text{HEARTBEAT\_INTERVAL}$ . A good rule of thumb is to have at least two or three heartbeats within one `NODE_TIMEOUT`.

## Adding or Removing Nodes While the Cluster is Running

You can reconfigure the cluster by adding or removing nodes while the cluster is up and running. The procedures are described in the chapter on “Cluster and Package Maintenance.”

## Verifying the Cluster Configuration

If you have edited an ASCII cluster configuration file, use the following command to verify the content of the file:

```
# cmcheckconf -v -C $SGCONF/clust1.config
```

This command checks the following:

- Network addresses and connections.
- Quorum server connection.
- All lock LUN device names on all nodes refer to the same physical disk area.
- One and only one lock LUN device is specified per node.
- A quorum server or lock LUN is configured, but not both.
- Uniqueness of names.
- Existence and permission of scripts specified in the command line.
- If all nodes specified are in the same heartbeat subnet.
- If you specify the wrong configuration filename.
- If all nodes can be accessed.
- No more than one `CLUSTER_NAME`, `HEARTBEAT_INTERVAL`, and `AUTO_START_TIMEOUT` are specified.
- The value for package run and halt script timeouts is less than 4294 seconds.
- The value for `HEARTBEAT_INTERVAL` is at least one second.
- The value for `NODE_TIMEOUT` is at least twice the value of `HEARTBEAT_INTERVAL`.
- The value for `AUTO_START_TIMEOUT` variables is  $\geq 0$ .
- Heartbeat network minimum requirement. The cluster must have one heartbeat LAN that is configured as a link aggregate of at least two interfaces.
- At least one `NODE_NAME` is specified.
- Each node is connected to each heartbeat network.
- All heartbeat networks are of the same type of LAN.

- The network interface device files specified are valid LAN device files.
- Other configuration parameters for the cluster and packages are valid.

If the cluster is online or the `cmcheckconf` command also verifies that all the conditions for the specific change in configuration have been met.

## Cluster Lock Configuration Messages

The `cmquerycl`, `cmcheckconf` and `cmapplyconf` commands will return errors if the cluster lock is not correctly configured. If there is no cluster lock in a cluster with two nodes, the following message is displayed in the ASCII configuration file:

```
# Warning: Neither a quorum server nor a lock lun was
specified.

# A Quorum Server or a lock lun is required for clusters of
only two nodes.
```

If you attempt to configure both a quorum server and a lock LUN, the following message appears on standard output when issuing the `cmcheckconf` or `cmapplyconf` command:

Duplicate cluster lock, line 55. Quorum Server already specified.

## Distributing the Binary Configuration File

After specifying all cluster parameters, you use the `cmapplyconf` command to apply the configuration. This action distributes the binary configuration file to all the nodes in the cluster. We recommend doing this separately *before* you configure packages (described in the next chapter). In this way, you can verify the quorum server, heartbeat networks, and other cluster-level operations by using the `cmviewcl` command on the running cluster. Before distributing the configuration, ensure that your security files permit copying among the cluster nodes. See “Preparing Your Systems” at the beginning of this chapter.

The following command distributes the binary configuration file:

```
# cmapplyconf -v -C $SGCONF/clust1.config
```

## Managing the Running Cluster

This section describes some approaches to routine management of the cluster. Additional tools and suggestions are found in Chapter 7, “Cluster and Package Maintenance.”

### Checking Cluster Operation with Serviceguard Manager

Serviceguard Manager lets you see all the nodes and packages within a cluster and displays their current status. Refer to the section on “Using Serviceguard Manager” in Chapter 7. The following are suggested using Serviceguard Manager:

- Ensure that all configured nodes are running.
- Check that all configured packages are running, and running on the correct nodes.
- Ensure that the settings on the property sheets for cluster, nodes, and packages are correct.

When you are sure the cluster is correctly configured, save a copy of the configuration data in a file for archival purposes. The data in this file can be compared with later versions of the cluster to understand the changes that are made over time.

### Checking Cluster Operation with Serviceguard Commands

Serviceguard also provides several commands for control of the cluster:

- `cmviewcl` checks status of the cluster and many of its components. A non-root user with the role of Monitor can run this command from a cluster node or see status information in Serviceguard Manager.
- `cmrunnode` is used to start a node. A non-root user with the role of Full Admin, can run this command from a cluster node or through Serviceguard Manager.

- `cmhaltnode` is used to manually stop a running node. (This command is also used by `shutdown(1m)`.) A non-root with the role of Full Admin can run this command from a cluster node or through Serviceguard Manager.
- `cmruncl` is used to manually start a stopped cluster. A non-root user with Full Admin access can run this command from a cluster node, or through Serviceguard Manager.
- `cmhaltcl` is used to manually stop a cluster. A non-root user with Full Admin access, can run this command from a cluster node or through Serviceguard Manager.

You can use these commands to test cluster operation, as in the following:

1. If the cluster is not already online, start it. From the Serviceguard Manager menu, choose Run Cluster. From the command line, use `cmruncl -v`.

By default, `cmruncl` will check the networks. Serviceguard will probe the actual network configuration with the network information in the cluster configuration. If you do not need this validation, use `cmruncl -v -w none` instead, to turn off validation and save time

2. When the cluster has started, make sure that cluster components are operating correctly. In Serviceguard Manager, open the cluster on the map or tree, and perhaps check its Properties. On the command line, use the `cmviewcl -v` command.

Make sure that all nodes and networks are functioning as expected. For more information, refer to the chapter on “Cluster and Package Maintenance.”

3. Verify that nodes leave and enter the cluster as expected using the following steps:
  - Halt the cluster. In Serviceguard Manager menu use Halt Cluster. On the command line, use the `cmhaltnode` command.
  - Check the cluster membership on the map or tree to verify that the node has left the cluster. In Serviceguard Manager, open the map or tree or Cluster Properties. On the command line, use the `cmviewcl` command.
  - Start the node. In Serviceguard Manager use the Run Node command. On the command line, use the `cmrunnode` command.

## Managing the Running Cluster

- To verify that the node has returned to operation, check the Serviceguard Manager map or tree, or use the `cmviewcl` command again.
4. Bring down the cluster. In Serviceguard Manager, use the Halt Cluster command. On the command line, use the `cmhaltcl -v -f` command.

Additional cluster testing is described in the “Troubleshooting” chapter. Refer to Appendix A for a complete list of Serviceguard commands. Refer to the Serviceguard Manager Help for a list of Serviceguard Administrative commands.

## Setting up Autostart Features

Automatic startup is the process in which each node individually joins a cluster; Serviceguard provides a startup script to control the startup process. If a cluster already exists, the node attempts to join it; if no cluster is running, the node attempts to form a cluster consisting of all configured nodes. Automatic cluster start is the preferred way to start a cluster. No action is required by the system administrator.

To enable automatic cluster start, set the flag `AUTOSTART_CMCLD` to 1 in the `$SGAUTOSTART` file (`$SGCONF/cmcluster.rc`) on each node in the cluster; the nodes will then join the cluster at boot time.

Here is an example of the `$SGAUTOSTART` file:

```
$SGAUTOSTART=/usr/local/cmcluster/conf/cmcluster.rc
# ***** CMCLUSTER *****

# Highly Available Cluster configuration
#
# @(#) $Revision: 82.2 $
#
#
# AUTOSTART_CMCLD
#
# Automatic startup is the process in which each node individually
# joins a cluster. If a cluster already exists, the node attempts
# to join it; if no cluster is running, the node attempts to form
# a cluster consisting of all configured nodes. Automatic cluster
# start is the preferred way to start a cluster. No action is
# required by the system administrator. If set to 1, the node will
# attempt to join/form its CM cluster automatically as described
# above. If set to 0, the node will not attempt to join its CM
# cluster.

AUTOSTART_CMCLD=1
```

## Changing the System Message

You may find it useful to modify the system's login message to include a statement such as the following:

```
This system is a node in a high availability cluster.  
Halting this system may cause applications and services to  
start up on another node in the cluster.
```

You might wish to include a list of all cluster nodes in this message, together with additional cluster-specific information.

The `/etc/motd` file may be customized to include cluster-related information.

## Managing a Single-Node Cluster

The number of nodes you will need for your HP Serviceguard cluster depends on the processing requirements of the applications you want to protect.

In a single-node cluster, a quorum server is not required, since there is no other node in the cluster. The output from the `cmquerycl` command omits the quorum server information area if there is only one node.

You still need to have redundant networks, but you do not need to specify any heartbeat LANs, since there is no other node to send heartbeats to. In the cluster configuration ASCII file, specify all LANs that you want Serviceguard to monitor. For LANs that already have IP addresses, specify them with the `STATIONARY_IP` keyword, rather than the `HEARTBEAT_IP` keyword.

### Single-Node Operation

Single-node operation occurs in a single-node cluster or in a multi-node cluster, following a situation where all but one node has failed, or where you have shut down all but one node, which will probably have applications running. As long as the HP Serviceguard daemon `cmclld` is active, other nodes can re-join the cluster at a later time.

If the HP Serviceguard daemon fails when in single-node operation, it will leave the single node up and your applications running. This is different from the loss of the HP Serviceguard daemon in a multi-node cluster, which halts the node with a TOC, and causes packages to be switched to adoptive nodes. It is not necessary to halt the single node in this scenario, since the application is still running, and no other node is



currently available for package switching. However, you should *not* try to restart HP Serviceguard, since data corruption might occur if another node were to attempt to start up a new instance of the application that is still running on the single node. Instead of restarting the cluster, choose an appropriate time to shutdown and reboot the node, which will allow the applications to shut down and then permit HP Serviceguard to restart the cluster after rebooting.

## Deleting the Cluster Configuration

You can delete a cluster configuration by issuing the `cmdeleteconf` command. The command prompts for a verification before deleting the files unless you use the `-f` option. You can only delete the configuration when the cluster is down. The action removes the binary configuration file from all the nodes in the cluster and resets all cluster-aware volume groups to be no longer cluster-aware.

---

### NOTE

The `cmdeleteconf` command removes only the cluster binary file `$SGCONF/cmclconfig`. It does *not* remove any other files from the `$SGCONF` directory.

---

Although the cluster must be halted, all nodes in the cluster should be powered up and accessible before you use the `cmdeleteconf` command. If a node is powered down, power it up and boot. If a node is inaccessible, you will see a list of inaccessible nodes together with the following message:

```
Checking current status
cmdeleteconf: Unable to reach node lpctest1.
WARNING: Once the unreachable node is up, cmdeleteconf
should be executed on the node to remove the configuration.
```

```
Delete cluster lpcluster anyway (y/[n])?
```

Reply Yes to remove the configuration. Later, if the inaccessible node becomes available, you should run the `cmdeleteconf` command on that node to remove the configuration file.



# 6

## Configuring Packages and Their Services

In addition to configuring the cluster, you need to identify the applications and services that you wish to group into packages. This chapter describes the following *package configuration* tasks:

- Creating the Package Configuration
- Writing the Package Control Script
- Verifying the Package Configuration
- Applying and Distributing the Configuration
- Creating a Disk Monitor Configuration

Each of these tasks is described in a separate section below.

In configuring your own packages, use data from the Package Configuration Worksheet described in the “Planning” chapter. Package configuration data from the worksheet becomes part of the binary cluster configuration file on all nodes in the cluster. The control script data from the worksheet goes into an executable package control script which runs specific applications and monitors their operation.

## Using Serviceguard Manager to Configure a Package

To configure a high availability package use the following steps on the configuration node (ftsys9). Serviceguard Manager configuration is available in Serviceguard Version A.11.16 and later.

1. Connect to a session server node that has Serviceguard version A.11.16 installed. Discover a cluster that has version A.11.16 or later. To begin configuration, you will be prompted to enter the root password for a node in the target cluster.
2. To create a package, select the cluster on the map or tree. From the Actions menu, choose Configuration -> Create Package. To modify, select the package itself and choose Configuration -> Modify Package <pkgname>.
3. Creating the Package Configuration and its Control Script: The interface will guide you through the steps. Online Help is available along the way. You can follow the suggestions below about configuring in stages, because many steps do not have to be done in sequence. The control script can be created automatically if you want.
4. Verifying the Package Configuration: Click the Check button. If you don't see the log window, open one from the View menu.
5. Distributing the Configuration: Click Apply. The binary configuration file will be created, then it and the generated control script will be distributed to the package's nodes. If you want, you can configure your control script yourself. This may be necessary if you do not use a standard control script. However, once you edit a control script yourself, Serviceguard Manager will never be able to see or modify it again. If you choose to edit the control script, you must also distribute it yourself.

## Creating the Package Configuration

The package configuration process defines a set of application services that are run by the package manager when a package starts up on a node in the cluster. The configuration also includes a prioritized list of cluster nodes on which the package can run together with definitions of the acceptable types of failover allowed for the package.

### Creating the Package Configuration File

Use the following procedure to create packages by editing and processing a package configuration file.

1. First, create a subdirectory for each package you are configuring in the `$SGCONF` directory:

```
# mkdir $SGCONF/pkg1
```

You can use any directory names you wish.

2. Next, generate a package configuration template for the package:

```
# cmmakepkg -p $SGCONF/pkg1/pkg1.config
```

You can use any file names you wish for the ASCII templates.

3. Edit these template files to specify package name, prioritized list of nodes, the location of the control script, and failover parameters for each package. Include the data recorded on the Package Configuration Worksheet.

## Configuring in Stages

It is recommended to configure packages on the cluster in stages, as follows:

1. Configure volume groups and mount points only.
2. Apply the configuration.
3. Distribute the control script to all nodes.
4. Run the package and ensure that it can be moved from node to node.
5. Halt the package.
6. Configure package IP addresses and application services in the control script.
7. Distribute the control script to all nodes.
8. Run the package and ensure that applications run as expected and that the package fails over correctly when services are disrupted.

## Package Configuration Template File

The following is a sample package configuration file template customized for a typical package.

```
# *****
# ***** HIGH AVAILABILITY PACKAGE CONFIGURATION FILE (template) *****
# *****
# ***** Note: This file MUST be edited before it can be used. *****
# * For complete details about package parameters and how to set them, *
# * consult the HP Serviceguard manuals. *
# *****

# Enter a name for this package. This name will be used to identify the
# package when viewing or manipulating it. It must be different from
# the other configured package names.

PACKAGE_NAME                                pkg1

# Enter the package type for this package.
# Currently the only valid value for PACKAGE TYPE is:
#
#     FAILOVER      package runs on one node at a time and if a failure
#                   occurs it can switch to an alternate node.
#
# Examples : PACKAGE_TYPE    FAILOVER (default)
```

```
#

PACKAGE_TYPE                                FAILOVER

# Enter the failover policy for this package. This policy will be used
# to select an adoptive node whenever the package needs to be started.
# The default policy unless otherwise specified is CONFIGURED_NODE.
# This policy will select nodes in priority order from the list of
# NODE_NAME entries specified below.

# The alternative policy is MIN_PACKAGE_NODE. This policy will select
# the node, from the list of NODE_NAME entries below, which is
# running the least number of packages at the time this package needs
# to start.

FAILOVER_POLICY                            CONFIGURED_NODE

# Enter the failback policy for this package. This policy will be used
# to determine what action to take when a package is not running on
# its primary node and its primary node is capable of running the
# package. The default policy unless otherwise specified is MANUAL.
# The MANUAL policy means no attempt will be made to move the package
# back to its primary node when it is running on an adoptive node.
#
# The alternative policy is AUTOMATIC. This policy will attempt to
# move the package back to its primary node whenever the primary node
# is capable of running the package.

FAILBACK_POLICY                            MANUAL

# Enter the names of the nodes configured for this package. Repeat
# this line as necessary for additional adoptive nodes.
#
# NOTE:   The order is relevant.
#         Put the second Adoptive Node after the first one.
#
# Example : NODE_NAME  original_node
#           NODE_NAME  adoptive_node
#
# If all nodes in the cluster are to be specified and order is not
# important, "NODE_NAME *" may be specified.
#
# Example: NODE_NAME  *

NODE_NAME                                  ftsys9
```

## Configuring Packages and Their Services

### Creating the Package Configuration

```
NODE_NAME                                ftsys10

# Enter the value for AUTO_RUN. Possible values are YES and NO.
# The default for AUTO_RUN is YES. When the cluster is started the
# package will be automatically started. In the event of a failure the
# package will be started on an adoptive node. Adjust as necessary.
#
# AUTO_RUN replaces obsolete PKG_SWITCHING_ENABLED.

AUTO_RUN                                YES

# Enter the value for NODE_FAIL_FAST_ENABLED.
# Possible values are YES and NO.
# The default for NODE_FAIL_FAST_ENABLED is NO. If set to YES,
# in the event of a failure, the cluster software will halt the node
# on which the package is running. All SYSTEM_MULTI_NODE packages must have
# NODE_FAIL_FAST_ENABLED set to YES. Adjust as necessary.

NODE_FAIL_FAST_ENABLED                  NO

# Enter the complete path for the run and halt scripts. In most cases
# the run script and halt script specified here will be the same script,
# the package control script generated by the cmmakepkg command. This
# control script handles the run(ning) and halt(ing) of the package.
# If the script has not completed by the specified timeout value,
# it will be terminated. The default for each script timeout is
# NO_TIMEOUT. Adjust the timeouts as necessary to permit full
# execution of each script.
# Note: The HALT_SCRIPT_TIMEOUT should be greater than the sum of
# all SERVICE_HALT_TIMEOUT values specified for all services.

RUN_SCRIPT                             /usr/local/cmcluster/conf/pkg1/control.sh
RUN_SCRIPT_TIMEOUT                      NO_TIMEOUT
HALT_SCRIPT                             /usr/local/cmcluster/conf/pkg1/control.sh
HALT_SCRIPT_TIMEOUT                    NO_TIMEOUT

# Enter the SERVICE_NAME, the SERVICE_FAIL_FAST_ENABLED and the
# SERVICE_HALT_TIMEOUT values for this package. Repeat these
# three lines as necessary for additional service names. All
# service names MUST correspond to the SERVICE_NAME entries in
# the package control script.
#
# The value for SERVICE_FAIL_FAST_ENABLED can be either YES or
# NO. If set to YES, in the event of a service failure, the
# cluster software will halt the node on which the service is
# running. If SERVICE_FAIL_FAST_ENABLED is not specified, the
```



```
# default will be NO.
#
# SERVICE_HALT_TIMEOUT is represented as a number of seconds.
# This timeout is used to determine the length of time (in
# seconds) the cluster software will wait for the service to
# halt before a SIGKILL signal is sent to force the termination
# of the service. In the event of a service halt, the cluster
# software will first send a SIGTERM signal to terminate the
# service. If the service does not halt, after waiting for the
# specified SERVICE_HALT_TIMEOUT, the cluster software will send
# out the SIGKILL signal to the service to force its termination.
# This timeout value should be large enough to allow all cleanup
# processes associated with the service to complete. If the
# SERVICE_HALT_TIMEOUT is not specified, a zero timeout will be
# assumed, meaning the cluster software will not wait at all
# before sending the SIGKILL signal to halt the service.
#
# Example: SERVICE_NAME                DB_SERVICE
#          SERVICE_FAIL_FAST_ENABLED    NO
#          SERVICE_HALT_TIMEOUT         300
#
# To configure a service, uncomment the following lines and
# fill in the values for all of the keywords.
#
#SERVICE_NAME                <service name>
#SERVICE_FAIL_FAST_ENABLED    <YES/NO>
#SERVICE_HALT_TIMEOUT         <number of seconds>

SERVICE_NAME                service1
SERVICE_FAIL_FAST_ENABLED    YES
SERVICE_HALT_TIMEOUT         300

# Enter the network subnet name that is to be monitored for this package.
# Repeat this line as necessary for additional subnet names. If any of
# the subnets defined goes down, the package will be switched to another
# node that is configured for this package and has all the defined subnets
# available.

SUBNET    15.16.168.0

# Access Control Policy Parameters.
#
# Three entries set the access control policy for the package:
# First line must be USER_NAME, second USER_HOST, and third USER_ROLE.
# Enter a value after each.
```

**Creating the Package Configuration**

```
#
# 1. USER_NAME can either be ANY_USER, or a maximum of
#    8 login names from the /etc/passwd file on user host.
# 2. USER_HOST is where the user can issue Serviceguard commands.
#    If using Serviceguard Manager, it is the COM server.
#    Choose one of these three values: ANY_SERVICEGUARD_NODE, or
#    (any) CLUSTER_MEMBER_NODE, or a specific node. For node,
#    use the official hostname from domain name server, and not
#    an IP addresses or fully qualified name.
# 3. USER_ROLE must be PACKAGE_ADMIN. This role grants permission
#    to MONITOR, plus for administrative commands for the package.
#
# These policies do not effect root users. Access Policies here
# should not conflict with policies defined in the cluster configuration file.
#
# Example: to configure a role for user john from node noir to
# administer the package, enter:
# USER_NAME  john
# USER_HOST  noir
# USER_ROLE  PACKAGE_ADMIN
```

Use the information on the Package Configuration worksheet to complete the file. Refer also to the comments on the configuration template for additional explanation of each parameter. You may include the following information:

- **FAILOVER\_POLICY.** Enter either **CONFIGURED\_NODE** or **MIN\_PACKAGE\_NODE**.
- **FAILBACK\_POLICY.** Enter either **MANUAL** or **AUTOMATIC**.
- **NODE\_NAME.** Enter the name of each node that is allowed to run the package on a separate line.
- **AUTO\_RUN.** Enter **YES** to allow the package to start on the first available node, or **NO** to keep the package from automatic startup.
- **RUN\_SCRIPT** and **HALT\_SCRIPT.** Specify the pathname of the package control script (described in the next section). No default is provided.
- If your package contains application services, enter the **SERVICE\_NAME**, **SERVICE\_FAIL\_FAST\_ENABLED** and **SERVICE\_HALT\_TIMEOUT.** values. Enter a group of these three for each service. You can configure no more than 30 services per package. The service name must also be defined in the package control script.

- If the package depends on monitored devices, be sure to include separate service entries (`SERVICE_NAME` DiskMond) for disk monitoring and include the `resclicant` command as the `SERVICE_CMD` in the control script.
- If your package has IP addresses associated with it, enter the `SUBNET`. This must be a subnet that is already specified in the cluster configuration.
- `NODE_FAIL_FAST_ENABLED` parameter. Enter YES or NO.

Access Control Policy is a new feature with Serviceguard version A.11.16. With it, you can allow non-root users to administer packages and clusters. Cluster-wide roles are defined in the cluster configuration file. Package-specific roles are defined in the package configuration file.

There must be no redundancy or conflict in roles. If there is, configuration will fail and you will get a message. It is a good idea, therefore, to look at the cluster configuration file (`cmgetconf`) before creating any roles in the package's file.

If a role is configured in the cluster, do not configure a role for the same username/hostnode in the package. The exception is wildcards. For example, if there were a `MONITOR` role for `ANY_USER` from `ftsys9` in the cluster configuration, and you created an `ADMIN` role for user Lee from `ftsys9` in the package configuration file, `ANY_USER` from `ftsys9` would have the role of `PACKAGE_ADMIN` for this policy. (This role already includes the `MONITOR` role for the entire cluster.)

This package configuration data is later combined with the cluster configuration data in the binary cluster configuration file.

## Writing the Package Control Script

The package control script contains all the information necessary to run all the services in the package, monitor them during operation, react to a failure, and halt the package when necessary.

Each package must have a separate control script, which must be executable. The control script is placed in the package directory and is given the same name as specified in the `RUN_SCRIPT` and `HALT_SCRIPT` parameters in the package ASCII configuration file. For security reasons, the control script must reside in a directory with the string `cmcluster` in the path.

The package control script template contains both the run instructions and the halt instructions for the package. You can use a single script for both run and halt operations, or, if you wish, you can create separate scripts. After completing the script, you must propagate it to all the nodes. For more information See “Copying Package Control Scripts with Linux commands” on page 205.

Use the following procedure to create a control script for the sample package *pkg1*.

First, generate a control script template:

```
# cmmakepkg -s $SGCONF/pkg1/pkg1.sh
```

You may customize the script, as described in the next section.

## Customizing the Package Control Script

Check the definitions and declarations at the beginning of the control script using the information in the Package Configuration worksheet. For more information see “Understanding the Location of Serviceguard Files” on page 122. You need to customize as follows:

- Update the `PATH` statement to reflect any required paths needed to start your services.
- Enter the names of LVM volume groups that will be activated.
- Add the names of LVM logical volumes and file systems that will be mounted on them. Specify the file system type (ext2 is the default; ext3 or reiserfs can also be used.).
- Define IP subnet and IP address pairs for your package.
- Add service name(s).
- Add service command(s).
- Add a service restart parameter, if desired.

---

### NOTE

Use care in defining service run commands. Each run command is executed by the control script in the following way:

- The `cmrunserv` command executes each run command and then monitors the process id of the process created by the run command.
- When the command started by `cmrunserv` exits, Serviceguard determines that a failure has occurred and takes appropriate action, which may include transferring the package to an adoptive node.
- If a run command is a shell script that runs some other command and then exits, Serviceguard will consider this normal exit as a *failure*.

To avoid problems in the execution of control scripts, ensure that each run command is the name of an actual service and that its process remains alive until the actual service stops.

---

If you need to define a set of run and halt operations in addition to the defaults, create functions for them in the sections under the heading `CUSTOMER DEFINED FUNCTIONS`.

## Optimizing for Large Numbers of Storage Units

Two variables are provided to allow performance improvement when employing a large number of file systems or storage groups. For more detail, see the comments in the control script template. They are:

- `CONCURRENT_FSCK_OPERATIONS`—defines a number of parallel fsck operations that will be carried out at package startup.
- `CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS`—defines a number of parallel mount operations during package startup and unmount operations during package shutdown.

## Configuring Disk Monitoring Services

Include a service entry for disk monitoring if the package depends on monitored disks. Use entries similar to the following:

```
SERVICE_NAME[0]="cmresserviced_Pkg1"  
SERVICE_CMD[0]="$SGBIN/cmresserviced /dev/sdd1"  
SERVICE_RESTART[0]=" "
```

The *service\_name* must be the same as the name coded in the package ASCII configuration file for the disk monitoring service. If *service\_name* is not included, `cmconfigres` will fail to find the monitored disks.

## Package Control Script Template File

The following is an excerpt from the package control script template file.

---

### NOTE

Comment out all references to *RAIDSTART*, *RAIDSTOP*, and *MD*. They are deprecated in Serviceguard for Linux but the package control script has not yet been updated to reflect this. Do not make any changes in the SOFTWARE RAID DATA REPLICATION section, as this is reserved for a possible future enhancement.

---

```
# *****
# *
# *      HIGH AVAILABILITY PACKAGE CONTROL SCRIPT(template)      *
# *
# *      Note: This file MUST be edited before it can be used.    *
# *
# *  You must have bash version 2 installed for this script to work *
# *  properly.  Also required is the arping utility available in the *
# *  iputils package.
# *
# *****

# The PACKAGE and NODE environment variables are set by
# Serviceguard at the time the control script is executed.
# Do not set these environment variables yourself!
# The package may fail to start or halt if the values for
# these environment variables are altered.

# Test to see if the shell is POSIX compliant. On RH6.2 /bin/bash (which
# is the default shell, ie. /bin/sh is sym linked to it) is version
# 1.x and does not support some of the features that this control
# script uses (specifically arrays). Bash version 2.x does support
```

### Writing the Package Control Script

```
# arrays and is included as /bin/bash2. We will first check to see of
# the shell that invoked us (/bin/bash) will work (in case someone
# changed it, if not we will use /bin/bash2.
#
# At SG installation time we checked to make sure
# that either /bin/bash will work with this control script or
# that /bin/bash2 is installed. The SG rpm would not install unless
# one of these conditions are true. On RH7.x the default /bin/bash
# shell is version 2 and thus will work fine.
sglinux[0]=1 >/dev/null 2>&1
if [ $? -gt 0 ]; then
    # not a valid shell
    # will invoking /bin/bash2
    exec /bin/bash2 -c "$0 $"
    exit 1
fi

. ${SGCONFFILE:=/etc/cmcluster.conf}

# UNCOMMENT the variables as you set them.

# Set PATH to reference the appropriate directories.
PATH=$SGSBIN:/bin:/sbin:/usr/bin:/usr/sbin

#
# REMOTE DATA REPLICATION DEFINITION
# Specify the remote data replication method.
# Leave the default, DATA_REP="none", if remote data replication is not used.
#
# If remote data replication is used for the package application data, set
```



```
# the variable DATA_REP to the data replication method. The current supported
# methods are "clx", "clxeva" and "xdcmd".
#
DATA_REP="none"

# SOFTWARE RAID DATA REPLICATION
# Specify the location of the Software RAID configuration file if "DATA_REP"
# is set to "xdcmd" or "XDCMD". This is the Extended Distance cluster
# Configuration file.
# A separate copy of this file will be used by each package. It is recommended
# that this file be kept in the package directory.
#
# For example:
# XDC_CONFIG_FILE="$SGCONF/pkg1/raid.conf"
#
# Set XDC_CONFIG_FILE as shown below if raid.conf file is in the
# package directory.
#
# XDC_CONFIG_FILE="${0%/*}/raid.conf"
#
#XDC_CONFIG_FILE=""

# MD (RAID) CONFIGURATION FILE
# Specify the configuration file that will be used to define
# the md raid devices for this package.
#
# NOTE: The multipath mechanisms that are supported for shared storage
# depend on the storage subsystem and the HBA driver in the
# configuration. Follow the documentation for those devices when setting
# up multipath. The MD driver was used with earlier versions of
```

### Writing the Package Control Script

```
# Serviceguard and may still be used by some storage system/HBA
# combinations. For that reason there are references to MD in the template
# files, worksheets, and other areas. Only use MD if your storage systems
# specifically calls out its use for multipath.
# If some other multipath mechanism is used (e.g. one built
# into an HBA driver), then references to MD, RAIDTAB, RAIDSTART, etc.
# should be commented out. If the references are in the comments, they
# can be ignored. References to MD devices, such as /dev/md0, should be
# replaced with the appropriate multipath device name.
#
# For example:
# RAIDTAB="/usr/local/cmcluster/conf/raidtab.sg"
#
#RAIDTAB=" "

# MD (RAID) COMMANDS
# Specify the method of activation and deactivation for md.
# Leave the default (RAIDSTART="raidstart", "RAIDSTOP="raidstop") if you want
# md to be started and stopped with default methods.
#
RAIDSTART="raidstart -c ${RAIDTAB}"
RAIDSTOP="raidstop -c ${RAIDTAB}"

# VOLUME GROUP ACTIVATION
# Specify the method of activation for volume groups.
# Leave the default ("VGCHANGE="vgchange -a y") if you want volume
# groups activated in default mode.
#
# VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a y"                                # Default
```

```
# VOLUME GROUPS
# Specify which volume groups are used by this package. Uncomment VG[0]=" "
# and fill in the name of your first volume group. You must begin with
# VG[0], and increment the list in sequence.
#
# For example, if this package uses your volume groups vg01 and vg02, enter:
#         VG[0]=vg01
#         VG[1]=vg02
#
# The volume group activation method is defined above. The filesystems
# associated with these volume groups are specified below. Ensure all the
# mds in the volume groups are included in the md activation above.
#
#VG[0]=" "

# MULTIPLE DEVICES
# Specify which md devices are used by this package. Uncomment MD[0]=" "
# and fill in the name of your first multiple device. You must begin
# with MD[0], and increment the list in sequence. The md devices are
# defined in the RAIDTAB file specified above.
#
# For example, if this package uses multiple devices md0 and md1,
# enter:
#         MD[0]=/dev/md0
#         MD[1]=/dev/md1
#
#MD[0]=" "

# FILESYSTEMS
```

**Writing the Package Control Script**

```

# The filesystems are defined as entries specifying the logical
# volume, the mount point, the file system type, the mount,
# umount and fsck options.
# Each filesystem will be fsck'd prior to being mounted.
# The filesystems will be mounted in the order specified during package
# startup and will be unmounted in reverse order during package
# shutdown.  Ensure that volume groups referenced by the logical volume
# definitions below are included in volume group definitions.
#
# Specify the filesystems which are used by this package. Uncomment
# LV[0]=""; FS[0]=""; FS_TYPE[0]=""; FS_MOUNT_OPT[0]="";
# FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]=" and fill in
# the name of your first logical volume, filesystem, type, mount,
# umount and fsck options for the file system.
# You must begin with LV[0], FS[0],
# FS_TYPE[0], FS_MOUNT_OPT[0], FS_UMOUNT_OPT[0], FS_FSCK_OPT[0]
# and increment the list in sequence.
#
# Valid types for FS_TYPE are 'ext2' and 'reiserfs'.
#
# For example, if this package uses the following:
# logical volume:  /dev/vg01/lvol1  /dev/vg01/lvol2
# mount point:    /pkg1a           /pkg1b
# filesystem type: ext2             reiserfs
# mount options:  read/write       read/write
#
# Then the following would be entered:
#
#     LV[0]=/dev/vg01/lvol1; FS[0]=/pkg1a; FS_TYPE[0]="ext2";
#
#     FS_MOUNT_OPT[0]="-o rw"; FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]="";
#

```

```
#      LV[1]=/dev/vg01/lvol2; FS[1]=/pkg1b; FS_TYPE[1]="reiserfs";
#      FS_MOUNT_OPT[1]="-o rw"; FS_UMOUNT_OPT[1]=""; FS_FSCK_OPT[1]="";
#
#LV[0]=""; FS[0]=""; FS_TYPE[0]=""; FS_MOUNT_OPT[0]=" "
#FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]=" "

# FILESYSTEM UNMOUNT COUNT
# Specify the number of unmount attempts for each filesystem during package
# shutdown. The default is set to 1.
#
FS_UMOUNT_COUNT=1

# FILESYSTEM MOUNT RETRY COUNT.
# Specify the number of mount retrys for each filesystem.
# The default is 0. During startup, if a mount point is busy
# and FS_MOUNT_RETRY_COUNT is 0, package startup will fail and
# the script will exit with 1. If a mount point is busy and
# FS_MOUNT_RETRY_COUNT is greater than 0, the script will attempt
# to kill the process(s) responsible for the busy mount point
# and then mount the file system. It will attempt to kill user and
# retry mount, for the number of times specified in FS_MOUNT_RETRY_COUNT.
# If the mount still fails after this number of attempts, the script
# will exit with 1.
# NOTE: If the FS_MOUNT_RETRY_COUNT > 0, the script will execute
# "fuser -kuv" to freeup busy mount point.
#
FS_MOUNT_RETRY_COUNT=0

#
# Configuring the concurrent operations below can be used to improve the
```

### Writing the Package Control Script

```
# performance for starting up or halting a package. The maximum value for
# each concurrent operation parameter is 1024. Set these values carefully.
# The performance could actually decrease if the values are set too high
# for the system resources available on your cluster nodes. Some examples
# of system resources that can affect the optimum number of concurrent
# operations are: number of CPUs, amount of available memory, the kernel
# configuration for nfile and nproc. In some cases, if you set the number
# of concurrent operations too high, the package may not be able to start
# or to halt. It is suggested that the number of concurrent operations be
# tuned carefully, increasing the values a little at a time and observing
# the effect on the performance, and the values should never be set to a
# value where the performance levels off or declines. Additionally, the
# values used should take into account the node with the least resources
# in the cluster, and how many other packages may be running on the node.
# For instance, if you tune the concurrent operations for a package so
# that it provides optimum performance for the package on a node while
# no other packages are running on that node, the package performance
# may be significantly reduced, or may even fail when other packages are
# already running on that node.
```

```
# CONCURRENT FSCK OPERATIONS
```

```
# Specify the number of concurrent fsck processes to allow during package
# startup. Setting this value to an appropriate number may improve the
# performance while checking a large number of file systems in the package.
# If the specified value is less than 1, the script defaults it to 1 and
# proceeds with a warning message in the package control script logfile.
```

```
CONCURRENT_FSCK_OPERATIONS=1
```

```
# CONCURRENT MOUNT AND UMOUNT OPERATIONS
```

```
# Specify the number of concurrent mounts and umounts to allow during
```

```
# package startup or shutdown.

# Setting this value to an appropriate number may improve the performance
# while mounting or un-mounting a large number of file systems in the package.
# If the specified value is less than 1, the script defaults it to 1 and
# proceeds with a warning message in the package control script logfile.
CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS=1


# IP ADDRESSES
# Specify the IP and Subnet address pairs which are used by this package.
# Uncomment IP[0]=" " and SUBNET[0]=" " and fill in the name of your first
# IP and subnet address. You must begin with IP[0] and SUBNET[0] and
# increment the list in sequence.
#
# For example, if this package uses an IP of 192.10.25.12 and a subnet of
# 192.10.25.0 enter:
#
#       IP[0]=192.10.25.12
#       SUBNET[0]=192.10.25.0 # (netmask=255.255.255.0)
#
# Hint: the subnet can be obtained by AND masking the IP address and the
#       netmask values from "ifconfig" command.
#
# IP/Subnet address pairs for each IP address you want to add to a subnet
# interface card. Must be set in pairs, even for IP addresses on the same
# subnet.
#
#IP[0]=" "
#SUBNET[0]=" "


# HA APPLICATION SERVER
# Enable or disable a High Availability application server that is used for
```

### Writing the Package Control Script

```
# this package. Some examples of the HA Servers are Network File System
# (NFS), Apache Web Server, and SAMBA (CIFS) Server.
#
# If you plan to use one of the HA server toolkits to run an application server,
# you need to set the HA_APP_SERVER value to either "pre-IP" or "post-IP" in
# order to enable this control script to check and run the Toolkit Interface
# Script (toolkit.sh) in the package directory. The interface script will call
# the toolkit main script to verify, start, and stop the server daemons.
#
# If you set the HA_APP_SERVER to "pre-IP", the application will be started
# BEFORE adding the package IP address(es) to the system. Application servers
# such as NFS and SAMBA are better to be started before the system provides
# external connections (activate package IP addresses). Therefore, at the time
# the clients connect to the system, the application server is
# ready for service.
#
# If you set the HA_APP_SERVER to "post-IP", the application will be started
# AFTER adding the package IP address(es) to the system. Application servers
# such as Apache Web Server will check the existing IP when the server starts.
# These applications will not be started if the IP has not been added to the
# system.
#
# Uncomment one the following lines as needed:
#
#HA_APP_SERVER="pre-IP"
#HA_APP_SERVER="post-IP"
#
# SERVICE NAMES AND COMMANDS.
# Specify the service name, command, and restart parameters which are
# used by this package. Uncomment SERVICE_NAME[0]="", SERVICE_CMD[0]="",
```



```
# SERVICE_RESTART[0]=" " and fill in the name of the first service, command,
# and restart parameters. You must begin with SERVICE_NAME[0], SERVICE_CMD[0],
# and SERVICE_RESTART[0] and increment the list in sequence.
#
# For example:
#
#     SERVICE_NAME[0]=cmresserviced_pkg1
#
#     SERVICE_CMD[0]="/usr/local/cmcluster/bin/cmresserviced /dev/md0"
#
#     SERVICE_RESTART[0]=" " # Will not restart the service.
#
#
#     SERVICE_NAME[1]=pkg1a
#
#     SERVICE_CMD[1]="/usr/bin/X11/xclock -display 192.10.25.54:0"
#
#     SERVICE_RESTART[1]=" " # Will not restart the service.
#
#
#     SERVICE_NAME[2]=pkg1c
#
#     SERVICE_CMD[2]="/bin/ping 127.0.0.1"
#
#     SERVICE_RESTART[2]="-R" # Will restart the service an infinite
#                               number of times.
#
#
# Note: No environmental variables will be passed to the command, this
# includes the PATH variable. Absolute path names are required for the
# service command definition. Default shell is /bin/sh.
#
#SERVICE_NAME[0]=" "
#SERVICE_CMD[0]=" "
#SERVICE_RESTART[0]=" "
```

The use of XFS is allowed and would be defined in the `FS_TYPE` field. Any options would be defined in the `FS_MOUNT_OPT` field.

The above excerpt from the control script shows the assignment of values to a set of variables. The remainder of the script uses these variables to control the package by executing Logical Volume Manager commands,

Linux commands, and Serviceguard commands including `cmrunserv`, `cmmodnet`, and `cmhaltserv`. Examine a copy of the control script template to see the flow of logic. Use the following command:

```
# cmmakepkg -s | more
```

The main function appears at the end of the script.

Note that individual variables are optional; you should include only as many as you need for proper package operation. For example, if your package does not need to activate a volume group, omit the `VG` variables; if the package does not use services, omit the corresponding `SERVICE_NAME`, `SERVICE_CMD`, and `SERVICE_RESTART` variables; and so on.

After customizing the script, distribute it to each node in the cluster using `scp`, `ftp`, or your favorite method of copying.

## Adding Customer Defined Functions to the Package Control Script

You can add additional shell commands to the package control script to be executed whenever the package starts or stops. Simply enter these commands in the `CUSTOMER DEFINED FUNCTIONS` area of the script. This gives you the ability to further customize the control script.

An example of this portion of the script is shown below, with the `date` and `echo` commands included to log starts and halts of the package to a special file.

```
# START OF CUSTOMER DEFINED FUNCTIONS

# This function is a place holder for customer defined functions.
# You should define all actions you want to happen here, before the
# service is started. You can create as many functions as you need.

function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some
command.
date >> /tmp/pkg1.datelog
echo 'Starting pkg1' >> /tmp/pkg1.datelog
test_return 51
}

# This function is a place holder for customer defined functions.
```

```
# You should define all actions you want to happen here, before the
service is
# halted.

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.
: # do nothing instruction, because a function must contain some
command.
date >> /tmp/pkg1.datelog
echo 'Halting pkg1' >> /tmp/pkg1.datelog
test_return 52
}
# END OF CUSTOMER DEFINED FUNCTIONS
```

### **Adding Serviceguard Commands in Customer Defined Functions**

You can add Serviceguard commands (such as `cmmodpkg`) in the Customer Defined Functions section of a package control script. However, these commands must not interact with the package itself. Additionally, if a Serviceguard command interacts with another package, then you need to check all packages with Serviceguard commands for the possibility of a command loop.

For instance, a command loop might occur under the following circumstances. Suppose `Pkg1` does a `cmmodpkg -d` of `Pkg2`, and `Pkg2` does a `cmmodpkg -d` of `Pkg1`. If both `Pkg1` and `Pkg2` start at the same time, `Pkg1` tries to `cmmodpkg Pkg2`. However, that `cmmodpkg` command has to wait for `Pkg2` startup to complete. `Pkg2` tries to `cmmodpkg Pkg1`, but `Pkg2` has to wait for `Pkg1` startup to complete, thereby causing a command loop.

To avoid this situation, it is a good idea to always specify a `RUN_SCRIPT_TIMEOUT` and a `HALT_SCRIPT_TIMEOUT` for all packages, especially packages that use Serviceguard commands in their control scripts. If a timeout is not specified and your configuration has a command loop as described above, inconsistent results can occur, including a hung cluster.

### **Adding or Removing Packages on a Running Cluster**

You can add or remove packages while the cluster is running, subject to the limit of `MAX_CONFIGURED_PACKAGES`. To add or remove packages online, refer to the chapter on “Cluster and Package Maintenance.”

## Verifying the Package Configuration

If you have edited an ASCII package configuration file, use the following command to verify the content of the file:

```
# cmcheckconf -v -P $SGCONF/pkg1/pkg1.config
```

Errors are displayed on the standard output. If necessary, edit the file to correct any errors, then run the command again until it completes without errors.

The `cmcheckconf` command checks the following:

- Package name is valid, and at least one `NODE_NAME` entry is included.
- There are no duplicate parameter entries.
- Values for parameters are within permitted ranges.
- Run and halt scripts exist on all nodes in the cluster and are executable.
- Run and halt script timeouts are less than 4294 seconds.
- Configured resources are available on cluster nodes.

---

## Applying and Distributing the Configuration

Use the `cmapplyconf` command to apply and distribute a binary cluster configuration file containing the package configuration among the nodes of the cluster. Example:

```
# cmapplyconf -v -C $SGCONF/cmcl.config -P \  
$SGCONF/pkg1/pkg1.config
```

The `cmapplyconf` command creates a binary cluster configuration database file and distributes it to all nodes in the cluster. This action ensures that the contents of the file are consistent across all nodes.

---

### NOTE

`cmcheckconf` and `cmapplyconf` must be used again any time changes are made to the cluster and package configuration files.

---

## Copying Package Control Scripts with Linux commands

Use Linux commands to copy package control scripts from the configuration node to the same pathname on all nodes which can possibly run the package. Use your favorite method of file transfer (e. g., `scp` or `ftp`). For example, from `ftsys9`, you can issue the `scp` command to copy the package control script to `ftsys10`:

```
# scp $SGCONF/pkg1/control.sh ftsys10:$SGCONF/pkg1/control.sh
```

---

### NOTE

If you use `ftp`, you have to set execute permission on the file after copying it.

---

## Testing Cluster and Package Operation

While configuring your Serviceguard cluster, it's a good idea to test that the various components of the cluster behave correctly in case of a failure. See the chapter “Troubleshooting Your Cluster” for an explanation of how to test that your cluster responds properly in the event of a package failure, a node failure, or a LAN failure.

## Creating a Disk Monitor Configuration

HP Serviceguard provides disk monitoring for the shared storage that is activated by packages in the cluster. The monitor daemon on each node tracks the status of all the disks that are flagged for monitoring on that node in the package configuration files and control scripts. In order to monitor disks, you first create monitor configuration files that include all the disks that you wish to track. The configuration is done separately for each node in the cluster, because each node monitors only the group of disks that can be activated on that node, and this group depends on which packages are allowed to run on the node.

To set up monitoring, first configure your packages, and include a service for monitoring in each package that has disks that you wish to track. Since service names must be unique across the cluster, you can use the package name in combination with a string that indicates monitoring. The following shows the entry as included in the package ASCII file for Pkg1:

```
SERVICE_NAME                cmresserviced_Pkg1
SERVICE_FAIL_FAST_ENABLED   YES
SERVICE_HALT_TIMEOUT        300
```

---

### CAUTION

Because of a limitation in LVM, `SERVICE_FAIL_FAST_ENABLED` must be set to `YES` to allow a package to fail over if it loses its storage; otherwise the package halt will not complete and the package will not fail over to another node.

---

The following shows the corresponding entry for this service in the control script:

```
SERVICE_NAME[0]="cmresserviced_Pkg1"
SERVICE_CMD[0]="cmresserviced /dev/sdd1 /dev/sde1"
SERVICE_RESTART[0]=" "
```

---

**NOTE**

The *SERVICE\_CMD* must include the string “**cmresserviced**” or the *cmconfigres* will fail to find the disks to be monitored. *cmconfigres* looks for *SERVICE\_CMD* entries including the text “**cmresserviced**” to determine which disks to monitor.

---

It is important to set *SERVICE\_RESTART* to an empty string (“”).

After setting up the package ASCII files and control scripts, use *cmcheckconf* and *cmapplyconf* to apply the package configuration, as usual. Then create a monitor configuration file using the *cmconfigres* command. This can be done for all nodes in the cluster at one time, or it can be done a node at a time. The first approach is easiest when first setting up the cluster, but the second approach may be more appropriate when modifying the monitoring configuration at a later time.

## Configuring All Disks for Monitoring

During initial cluster setup, to create a monitoring configuration that includes all disks for all packages that contain a monitoring service, use the following command:

```
# cmconfigres create
```

This command builds the *\$SGCONF/cmresmond\_config.xml* file on each node in the cluster. The command operates by examining a node at a time and scanning all the package control scripts for the packages that might at some time be running on that node. Note that the content of the *\$SGCONF/cmresmond\_config.xml* files is different from node to node if the list of packages that can run on each node is different from other nodes.

The following is a specimen of a *cmresmond\_config.xml* file:

```
<?xml version="1.0" encoding="UTF-8"?>
<resourceMonitorServerConfig
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ResMonServerConfig.xsd">
  <runDirectory>/usr/local/cmcluster/run</runDirectory>
  <htmlRoot>/usr/local/cmcluster/htdocs</htmlRoot>

  <logFile>/usr/local/cmcluster/log/ResMonServer.log</logFile>
  <logLevel>0</logLevel>
  <port>3542</port>
```

**Creating a Disk Monitor Configuration**

```

<serverPriority>
  <schedClass>normal</schedClass>
  <schedPriority>1</schedPriority>
</serverPriority>
<resourceConfigCategoryList>
<resourceCategory>
  <categoryName>Pkg1</categoryName>
  <agent>/usr/local/cmcluster/bin/cmcheckdisk</agent>
  <agentTimeout>60</agentTimeout>
  <pollingInterval>60</pollingInterval>
  <resourceConfig>
    <name>/dev/sdd1</name>
  </resourceConfig>
  <resourceConfig>
    <name>/dev/sde1</name>
  </resourceConfig>
</resourceCategory>
<resourceCategory>
  <categoryName>Pkg2</categoryName>
  <agent>/usr/local/cmcluster/bin/cmcheckdisk</agent>
  <agentTimeout>60</agentTimeout>
  <pollingInterval>60</pollingInterval>
  <resourceConfig>
    <name>/dev/sdf1</name>
  </resourceConfig>
</resourceCategory>
</resourceConfigCategoryList>
</resourceMonitorServerConfig>

```

The `cmconfigres create` command can be issued when the cluster is running, but packages that activate monitored disks cannot be running because it is not possible for them to stay up if the `cmresserviced` command fails. Also if the configuration is not complete, this service fails.

After configuring monitoring, run `cmresmond --start` before starting the cluster. Failure to do this will result in any disk monitor dependent packages to fail at run time. The `cmresmond --start` command must be executed on each node in order to start the disk monitoring daemon. After initial configuration, `cmresmond` will start at each reboot.

To change an existing configuration, you must first delete the existing configuration before you can create a new one. The `cmconfigres create` command will not succeed if a configuration already exists. When the server configuration is changed (via `cmconfigres`) and the disk monitor



is already running, `cmresmond --restart` should be run to apply this new configuration. Use the following commands to complete the change and active the new configuration:

```
# cmconfigres --delete
# cmconfigres --create
# cmresmond --restart
```

## Configuring Disks on a Single Node for Monitoring

An alternate method for configuring monitoring is to use the `sync` option to configure monitor requests for a single node. To avoid unexpected results, this method should be used when a disk is added to a package or when a new package is added that will have disk monitoring. It is not necessary to halt the cluster or other monitored packages that are not being changed. You may wish to configure the monitoring first on a node that is not currently running the package, then test the configuration by failing the package over to the node you have configured. If the configuration is successful, the package will start with monitoring active, after which you can do the configuration on the other nodes that can run the package.

---

### NOTE

Using the `sync` option does not affect the monitoring configuration of any package that is not being changed on the node.

---

The following is an example using the `sync` option to configure monitoring of the disks that might at some time be activated on the current node:

```
# cmconfigres sync
# cmresmond --reload
```

This will create or modify the `$SGCONF/cmresmond_config.xml` file and ensure that the `cmresmond reload` uses the new configuration file on the current node only. It is important that the commands be repeated on all nodes where the new or changed package might run. (If `NODE_NAME *` is specified in the package ASCII file, this means all nodes.)

## **Creating a Disk Monitor Configuration**

See “Viewing Status of Monitored Disks” on page 240 for information about viewing the status of disks in the cluster. More information about disk monitoring is provided in “Resource Monitor Daemon: cmresmond” on page 35, “Disk Monitor Services” on page 62, and “Monitoring Disks” on page 80.

# 7

## Cluster and Package Maintenance

This chapter describes the `cmviewcl` command, then shows how to start and halt a cluster or an individual node, how to perform permanent reconfiguration, and how to start, halt, move, and modify packages during routine maintenance of the cluster. Topics are as follows:

- Reviewing Cluster and Package States with the `cmviewcl` Command
- Using Serviceguard Manager
- Viewing Status of Monitored Disks
- Managing the Cluster and Nodes
- Reconfiguring a Running Cluster
- Managing Packages and Services
- Reconfiguring a Package on a Running Cluster
- Responding to Cluster Events
- Single-Node Operation
- Removing Serviceguard from a System

All administration tasks can be carried out using Linux commands.

---

## Reviewing Cluster and Package States with the `cmviewcl` Command

Information about cluster status is stored in the status database, which is maintained on each individual node in the cluster. You can display information contained in this database by issuing the `cmviewcl` command:

```
# cmviewcl -v
```

You can issue the `cmviewcl` command with non-root access: In clusters with Serviceguard version A.11.16 create a Monitor role in the cluster configuration file. In earlier versions, add a non-root pair to the `cmclnodelist` file (`<nodename> <nonrootuser>`).

The `cmviewcl` command, when issued with the `-v` option, displays information about all the nodes and packages in a running cluster, together with the settings of parameters that determine failover behavior.

---

### TIP

Some commands take longer to complete in large configurations. In particular, you can expect Serviceguard's CPU usage to increase during `cmviewcl -v` as the number of packages and services increases.

---

See the man page for a detailed description of other `cmviewcl` options.

## Types of Cluster and Package States

A cluster or its component nodes may be in several different states at different points in time. The following sections describe many of the common conditions the cluster or package may be in.

### Cluster Status

The *status* of a cluster may be one of the following:

- **Up.** At least one node has a running cluster daemon, and reconfiguration is not taking place.
- **Down.** No cluster daemons are running on any cluster node.
- **Starting.** The cluster is in the process of determining its active membership. At least one cluster daemon is running.
- **Unknown.** The node on which the `cmviewcl` command is issued cannot communicate with other nodes in the cluster.

### Node Status and State

The *status* of a node is either up (*active* as a member of the cluster) or down (*inactive* in the cluster), depending on whether its cluster daemon is running or not. Note that a node might be down from the cluster perspective, but still up and running Linux.

A node may also be in one of the following states:

- **Failed.** A node never sees itself in this state. Other active members of the cluster will see a node in this state if that node was in an active cluster, but is no longer, and is not halted.
- **Reforming.** A node is in this state when the cluster is re-forming. The node is currently running the protocols which ensure that all nodes agree to the new membership of an active cluster. If agreement is reached, the status database is updated to reflect the new cluster membership.
- **Running.** A node in this state has completed all required activity for the last re-formation and is operating normally.
- **Halted.** A node never sees itself in this state. Other nodes will see it in this state after the node has gracefully left the active cluster, for instance with a `cmhaltnode` command.

- **Unknown.** A node never sees itself in this state. Other nodes assign a node this state if it has never been an active cluster member.

### **Quorum Server Status and State**

The *status* of the quorum server can be one of the following:

- **Up.** The quorum server is active.
- **Down.** The quorum server is not active.

The *state* of the quorum server can be one of the following:

- **Running.** Quorum services are active and available.
- **Unsupported Version.** There is a version mismatch between Serviceguard and Quorum Server.
- **Access Denied.** The Serviceguard node is not authorized to access the quorum server system.
- **Unknown.** We cannot determine whether the quorum server is up or down. This may happen when the quorum server cannot be reached from the current node.

### **Package Status and State**

The *status* of a package can be one of the following:

- **Up.** The package control script is active.
- **Down.** The package control script is not active.
- **Unknown.**

The *state* of the package can be one of the following:

- **Starting.** The start instructions in the control script are being run.
- **Running.** Services are active and being monitored.
- **Halting.** The halt instructions in the control script are being run.

## Package Switching Attributes

Packages also have the following switching attributes:

- `AUTO_RUN`. Enabled means that the package can switch to another node in the event of failure.
- Node Switching. Enabled means that the package can switch to the referenced node. Disabled means that the package cannot switch to the specified node until the node is enabled for the package using the `cmmodpkg` command.

Every package is marked Enabled or Disabled for each node that is either a primary or adoptive node for the package.

## Service Status

Services have only status, as follows:

- Up. The service is being monitored.
- Down. The service is not running. It may have halted or failed.
- Uninitialized. The service is included in the cluster configuration, but it was not started with a run command in the control script.
- Unknown.

## Network Status

The network interfaces have only status, as follows:

- Up.
- Down.
- Unknown. We cannot determine whether the interface is up or down. This can happen when the cluster is down. A PRIMARY interface has this status.

### Failover and Failback Policies

Packages can be configured with one of two values for the `FAILOVER_POLICY` parameter:

- `CONFIGURED_NODE`. The package fails over to the next node in the node list in the package configuration file.
- `MIN_PACKAGE_NODE`. The package fails over to the node in the cluster with the fewest running packages on it.

Packages can also be configured with one of two values for the `FAILBACK_POLICY` parameter:

- `AUTOMATIC`. With this setting, a package, following a failover, returns to its primary node when the primary node becomes available again.
- `MANUAL`. With this setting, a package, following a failover, must be moved back to its original node by a system administrator.

Failover and failback policies are displayed in the output of the `cmviewcl -v` command.



## Examples of Cluster and Package States

The following sample output from the `cmviewcl -v` command shows status for the cluster in the sample configuration.

### Normal Running Status

Everything is running normally; both nodes in the cluster are running, and the packages are in their primary locations.

```
CLUSTER      STATUS
example      up

  NODE      STATUS      STATE
  ftsys9    up          running

Network_Parameters:
INTERFACE    STATUS      NAME
PRIMARY      up          eth0
PRIMARY      up          eth1

PACKAGE      STATUS      STATE      AUTO_RUN      NODE
pkg1         up          running    enabled        ftsys9

Policy_Parameters:
POLICY_NAME    CONFIGURED_VALUE
Failover       configured_node
Failback       manual

Script_Parameters:
ITEM           STATUS      MAX_RESTARTS  RESTARTS      NAME
Service        up          0             0             service1
Subnet          up          0             0             15.13.168.0

Node_Switching_Parameters:
NODE_TYPE      STATUS      SWITCHING      NAME
Primary        up          enabled        ftsys9        (current)
Alternate       up          enabled        ftsys10

NODE      STATUS      STATE
ftsys10   up          running

Network_Parameters:
INTERFACE    STATUS      NAME
PRIMARY      up          eth0
PRIMARY      up          eth1

PACKAGE      STATUS      STATE      AUTO_RUN      NODE
pkg2         up          running    enabled        ftsys10

Policy_Parameters:
POLICY_NAME    CONFIGURED_VALUE
Failover       configured_node
Failback       manual

Script_Parameters:
```

**Reviewing Cluster and Package States with the `cmviewcl` Command**

ITEM	STATUS	MAX_RESTARTS	RESTARTS	NAME
Service	up	0	0	service2
Subnet	up	0	0	15.13.168.0

**Node\_Switching\_Parameters:**

NODE_TYPE	STATUS	SWITCHING	NAME	
Primary	up	enabled	ftsys10	(current)
Alternate	up	enabled	ftsys9	

**Quorum Server Status**

If the cluster is using a quorum server for tie-breaking services, the display shows the server name, state and status following the entry for each node, as in the following excerpt from the output of `cmviewcl -v`:

CLUSTER	STATUS
example	up

NODE	STATUS	STATE
ftsys9	up	running

**Quorum Server Status:**

NAME	STATUS	STATE
lp-qs	up	running

...

NODE	STATUS	STATE
ftsys10	up	running

**Quorum Server Status:**

NAME	STATUS	STATE
lp-qs	up	running

## Status After Halting a Package

After halting pkg2 with the cmhaltpkg command, the output of cmviewcl-v is as follows:

```

CLUSTER      STATUS
example      up

      NODE      STATUS      STATE
      ftsys9    up          running

      Network_Parameters:
      INTERFACE  STATUS      NAME
      PRIMARY   up          eth0
      PRIMARY   up          eth1

      PACKAGE    STATUS      STATE      AUTO_RUN  NODE
      pkg1       up          running    enabled   ftsys9

      Policy_Parameters:
      POLICY_NAME  CONFIGURED_VALUE
      Failover     configured_node
      Failback     manual

      Script_Parameters:
      ITEM          STATUS  MAX_RESTARTS  RESTARTS  NAME
      Service       up          0             0         service1
      Subnet        up          0             0         15.13.168.0

      Node_Switching_Parameters:
      NODE_TYPE    STATUS      SWITCHING  NAME
      Primary      up          enabled    ftsys9    (current)
      Alternate    up          enabled    ftsys10

      NODE      STATUS      STATE
      ftsys10    up          running

      Network_Parameters:
      INTERFACE  STATUS      NAME
      PRIMARY   up          eth0
      PRIMARY   up          eth1

UNOWNED_PACKAGES

      PACKAGE    STATUS      STATE      AUTO_RUN  NODE
      pkg2       down        unowned    disabled   unowned

      Policy_Parameters:
      POLICY_NAME  CONFIGURED_VALUE
      Failover     configured_node
      Failback     manual

      Script_Parameters:
      ITEM          STATUS  NODE_NAME  NAME
      Service       up          0          0         service2
      Subnet        up          0          0         15.13.168.0

      Node_Switching_Parameters:
      NODE_TYPE    STATUS      SWITCHING  NAME
  
```

Reviewing Cluster and Package States with the `cmviewcl` Command

Primary	up	enabled	ftsys10
Alternate	up	enabled	ftsys9

Pkg2 now has the status “down”, and it is shown as in the unowned state, with package switching disabled. Note that switching is enabled for both nodes, however. This means that once global switching is re-enabled for the package, it will attempt to start up on the primary node.

## Status After Moving the Package to Another Node

After issuing the following command:

```
# cmrunpkg -n ftsys9 pkg2
```

the output of the `cmviewcl -v` command is as follows:

```
CLUSTER      STATUS
example      up

NODE          STATUS      STATE
ftsys9        up          running

Network_Parameters:
INTERFACE     STATUS      NAME
PRIMARY       up          eth0
PRIMARY       up          eth1

PACKAGE       STATUS      STATE      AUTO_RUN  NODE
pkg1          up          running    enabled   ftsys9

Policy_Parameters:
POLICY_NAME    CONFIGURED_VALUE
Failover       configured_node
Failback       manual

Script_Parameters:
ITEM           STATUS      MAX_RESTARTS  RESTARTS    NAME
Service        up          0             0           service1
Subnet          up          0             0           15.13.168.0

Node_Switching_Parameters:
NODE_TYPE      STATUS      SWITCHING     NAME
Primary        up          enabled       ftsys9      (current)
Alternate       up          enabled       ftsys10

PACKAGE       STATUS      STATE      AUTO_RUN  NODE
pkg2          up          running    disabled   ftsys9

Policy_Parameters:
POLICY_NAME    CONFIGURED_VALUE
Failover       configured_node
Failback       manual

Script_Parameters:
ITEM           STATUS      NAME          MAX_RESTARTS  RESTARTS
Service        up          0             0             service2
```

```

Subnet      up              0          0      15.13.168.0

Node_Switching_Parameters:
NODE_TYPE   STATUS      SWITCHING  NAME
Primary     up          enabled    ftsys10
Alternate   up          enabled    ftsys9    (current)

NODE        STATUS      STATE
ftsys10     up          running

Network_Parameters:
INTERFACE   STATUS      NAME
PRIMARY     up          eth0
PRIMARY     up          eth1

```

Now `pkg2` is running on node `ftsys9`. Note that it is still disabled from switching.

### Status After Package Switching is Enabled

The following command changes package status back to Auto Run Enabled:

```
# cmmodpkg -e pkg2
```

The output of the `cmviewcl` command is now as follows:

```

CLUSTER      STATUS
example      up

      NODE        STATUS      STATE
      ftsys9      up          running

      PACKAGE     STATUS      STATE      AUTO_RUN  NODE
      pkg1        up          running    enabled    ftsys9
      pkg2        up          running    enabled    ftsys9

      NODE        STATUS      STATE
      ftsys10     up          running

```

Both packages are now running on `ftsys9` and `pkg2` is enabled for switching. `Ftsys10` is running the daemon and no packages are running on `ftsys10`.

**Status After Halting a Node**

After halting *ftsys10*, with the following command:

```
# cmhaltnode ftsys10
```

the output of `cmviewcl` is as follows on *ftsys9*:

CLUSTER	STATUS			
example	up			

NODE	STATUS	STATE		
ftsys9	up	running		

PACKAGE	STATUS	STATE	AUTO_RUN	NODE
pkg1	up	running	enabled	ftsys9
pkg2	up	running	enabled	ftsys9

NODE	STATUS	STATE		
ftsys10	down	halted		

This output is seen on both *ftsys9* and *ftsys10*.

## Viewing Data on Unowned Packages

The following example shows packages that are currently unowned, that is, not running on any configured node.

UNOWNED\_PACKAGES

PACKAGE	STATUS	STATE	AUTO_RUN	NODE
PKG3	down	halted	enabled	unowned

Policy\_Parameters:

POLICY_NAME	CONFIGURED_VALUE
Failover	min_package_node
Failback	automatic

Script\_Parameters:

ITEM	STATUS	NODE_NAME	NAME
Subnet	up	manx	192.8.15.0
Subnet	up	burmese	192.8.15.0
Subnet	up	tabby	192.8.15.0
Subnet	up	persian	192.8.15.0

Node\_Switching\_Parameters:

NODE_TYPE	STATUS	SWITCHING	NAME
Primary	up	enabled	manx
Alternate	up	enabled	burmese
Alternate	up	enabled	tabby
Alternate	up	enabled	persian

## Using Serviceguard Manager

Serviceguard Manager is a graphical user interface for managing Serviceguard clusters. Use Serviceguard Manager to display data from running clusters, or optionally to view saved data files. Serviceguard Manager runs on HP-UX, Linux, and Windows systems. From there, you can view Serviceguard clusters on HP-UX or Linux. Options can be selected through dialog boxes or at the command line.

Refer to the latest *Serviceguard Manager Release Notes* for additional information about installing Serviceguard Manager. Serviceguard Manager can be downloaded free from <http://www.hp.com/go/softwaredepot>

## How Serviceguard Manager Works

Serviceguard Manager gets cluster data from an Object Manager daemon which establishes connections with all managed cluster nodes to obtain configuration data. The Object Manager daemon may be located on a different system from Serviceguard Manager itself.

You use Serviceguard Manager by first logging on to the object manager server and then specifying the cluster or nodes that should be displayed. Then the object manager will obtain data from the specified cluster or nodes, if security requirements are met. For the object manager connection to be successful, you must include the name of the Object Manager host in the security files (`/etc/cmclnodelist` or `.rhosts`) of each node you wish to display in Serviceguard Manager. Refer to the section “Preparing Your Systems” in Chapter 5 for information on how to edit this file.

---

### TIP

The Serviceguard Manager GUI can now be used to configure clusters and packages.

---



## Running Serviceguard Manager with a Command

To start Serviceguard Manager from the command line, either in HP-UX, Linux, or from a Windows NT DOS window, use the following syntax:

```
sgmgr [-s server [-l username [-p password]] | -f filename ]
      [-c clustername...]
```

Table 7-1 shows the use of the various options.

**Table 7-1**

**Serviceguard Manager Command Line Options**

Option	Description
<i>-s server</i>	Specifies the name of the server on which the Object Manager is running. This Object Manager gathers data about the cluster or clusters you wish to observe. You can choose this option or <i>-f filename</i> , but not both.
<i>-l username</i>	Login name on the server specified with the <i>-s server</i> option.
<i>-p password</i>	Password for the username and server specified above. You can only use this option if you specify a server name. If you include a server name, but omit this option, you will be prompted for a login name and password.
<i>-c clustername...</i>	Name of the cluster you wish to see displayed.
<i>-f filename.</i>	Name of a file containing stored data that you wish to view instead of connecting to a server. You can choose either this option or <i>-s server</i> , but not both.

Refer to the *sgmgr* (1M) manpage for additional syntax information.

## Starting with a Specific Cluster

Use the following procedures to start Serviceguard Manager to display a specific cluster. This approach is recommended as the fastest way to bring up the display:

- From the HP-UX management station, enter the `sgmgr` command from a terminal window in the install directory, typically:  

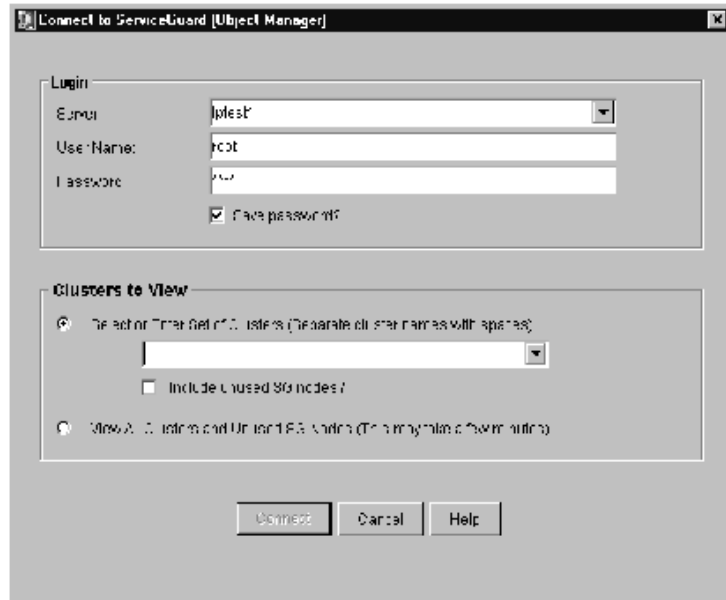
```
# /opt/sgmgr/bin/sgmgr -c <clustername>
```
- From the Red Hat management station, enter the `sgmgr` command from a terminal window in the install directory, typically:  

```
# /usr/local/sgmgr/bin/sgmgr -c <clustername>
```
- From the SUSE Linux management station, enter the `sgmgr` command from a terminal window in the install directory, typically:  

```
# export DISPLAY=mydisplay:0.0  
# /opt/sgmgr/bin/sgmgr -c <clustername>
```
- From the Windows management station:
  - Right-click the Serviceguard Manager icon on your desktop.
  - Click on Properties
  - Add the string “-c <clustername>” to the end of the command line.
  - Close the Properties dialog box.
  - Double click the Serviceguard Manager icon.

When started, the Serviceguard Manager splash screen shows program initialization. Then you will see the Object Manager login screen displayed in Figure 7-1.

**Figure 7-1**      **Object Manager Login Screen**



Proceed to the section below entitled “Connecting to an Object Manager.”

## Starting Serviceguard Manager without a Specific Cluster

Use the following procedures to start Serviceguard Manager without specifying a particular cluster. This approach will search for clusters throughout your subnet; if many clusters are configured, the search can be time-consuming.

- From the HP-UX management station, enter the `sgmgr` command from a terminal window in the install directory, typically:

```
# export DISPLAY=mydisplay:0.0
# /opt/sgmgr/bin/sgmgr
```

- From the Red Hat management station, enter the `sgmgr` command from a terminal window in the install directory, typically:

```
# export DISPLAY=mydisplay:0.0
# /usr/local/sgmgr/bin/sgmgr
```

- From the SuSE management station, enter the `sgmgr` command from a terminal window in the install directory, typically:

```
# export DISPLAY=mydisplay:0.0
# /opt/sgmgr/bin/sgmgr
```

- From the Windows management station, choose one of the following:

- Double-click the `sgmgr` icon on your desktop.
- From the PC Start menu, choose:  
Programs -> Serviceguard Manager -> `sgmgr`
- From “My Computer”

1. Go to:

```
C:\Program Files\Hewlett-Packard\Serviceguard
Manager\bin
```

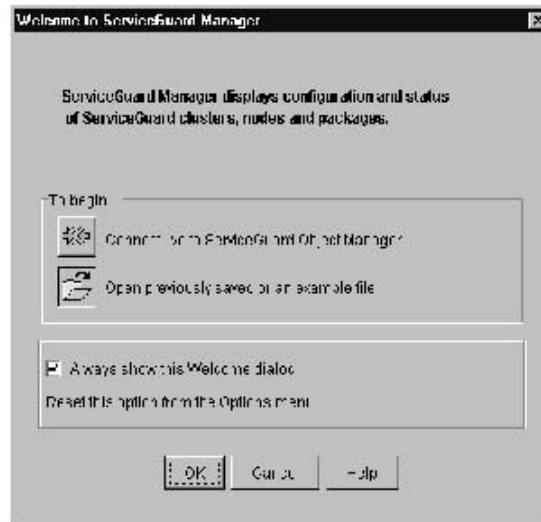
2. Double-click the `sgmgr.exe` symbol.

- From a DOS window, use the following commands:

```
C:> CD C:\Program Files\Hewlett-Packard\Serviceguard
Manager\bin
C:> sgmgr
```

When started, the Serviceguard Manager splash screen appears briefly. Then you see the opening screen displayed in Figure 7-2.

**Figure 7-2**      **Serviceguard Manager Opening Screen**



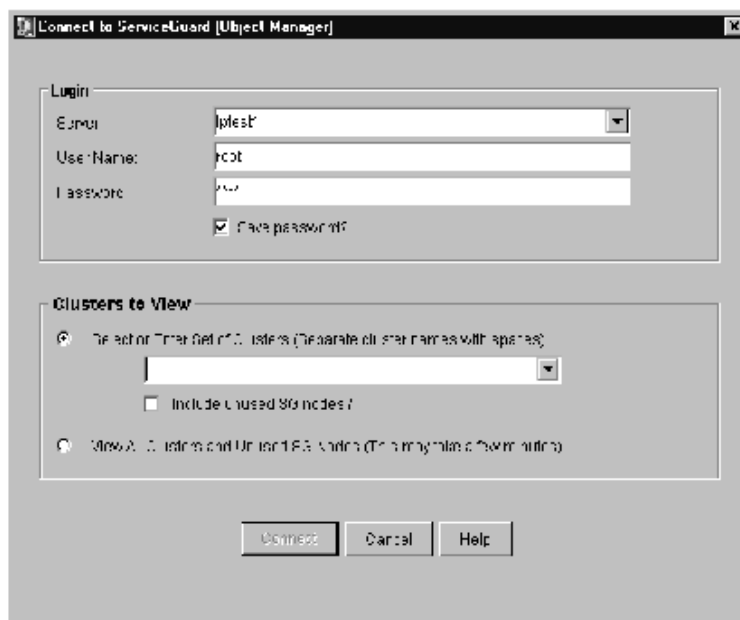
This screen lets you choose one of the following:

- to connect to Serviceguard Object Manager to view data from a running cluster
- to open a saved file to view data previously saved.

## Connecting to an Object Manager

The Serviceguard Object Manager daemon on a cluster node provides information about running clusters to Serviceguard Manager. If you want to view data about running clusters, you need to establish a connection to an Object Manager by logging on to a node running HP Serviceguard 11.12 or later with the Object Manager daemon running. The login screen is shown in Figure 7-3.

**Figure 7-3** Object Manager Login Screen



Enter the name of an object manager node, together with a valid user name and password. Valid users appear in the `/$.SGCONF/cmclnodelist` or `.rhosts` file on the Object Manager server.

---

**NOTE**

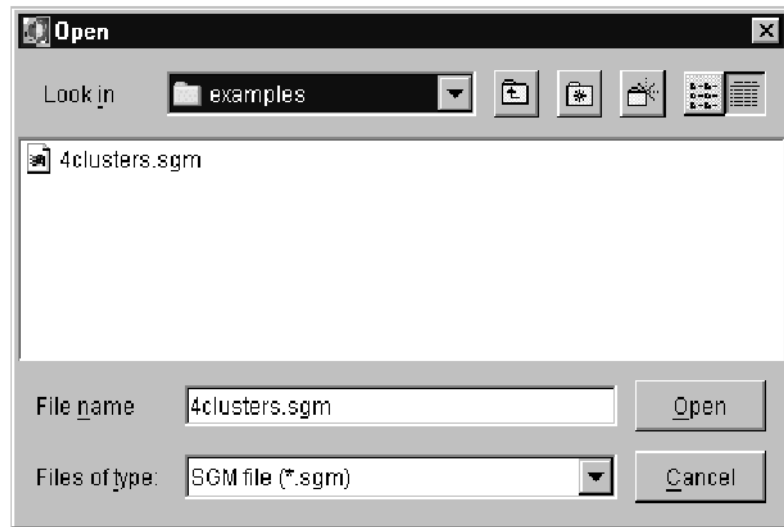
You need to log in as `root` in order to do administrative functions.

---

## Opening a Saved File with Cluster Data

Choosing the second option from the opening screen lets you open a file of saved data. You will see a list of saved files from which to choose, as shown in Figure 7-4.

**Figure 7-4** Serviceguard Manager Open File Screen

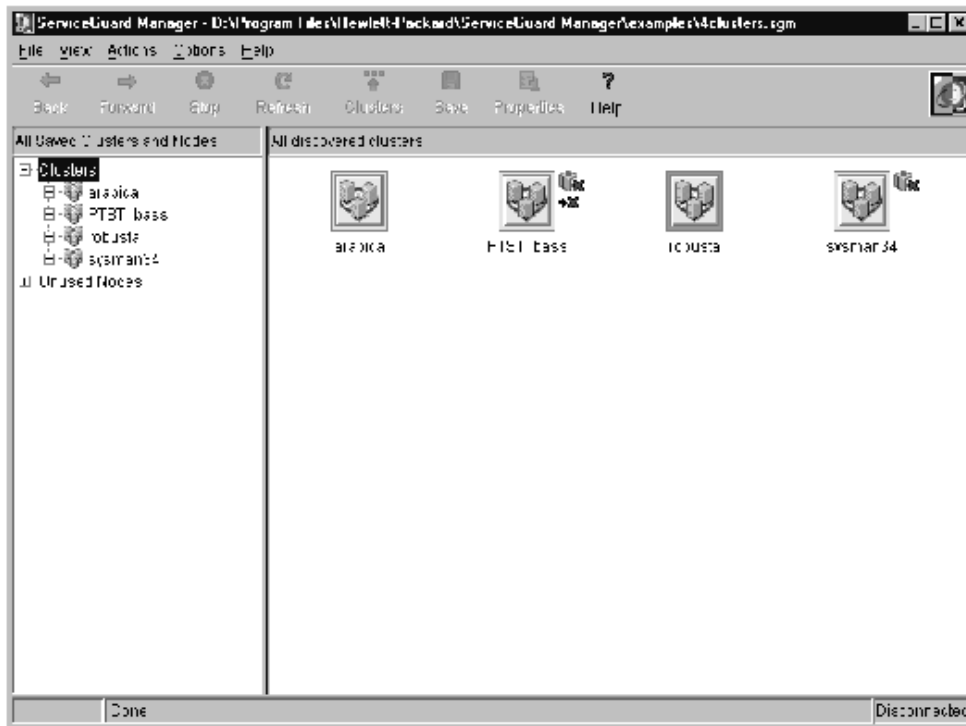


Choose the file you wish, then click “Open.”

## Viewing Cluster Data

Whether you are connecting to an Object Manager node or displaying data from a saved file, you will see an initial map and hierarchical tree display. This screen is shown in Figure 7-5.

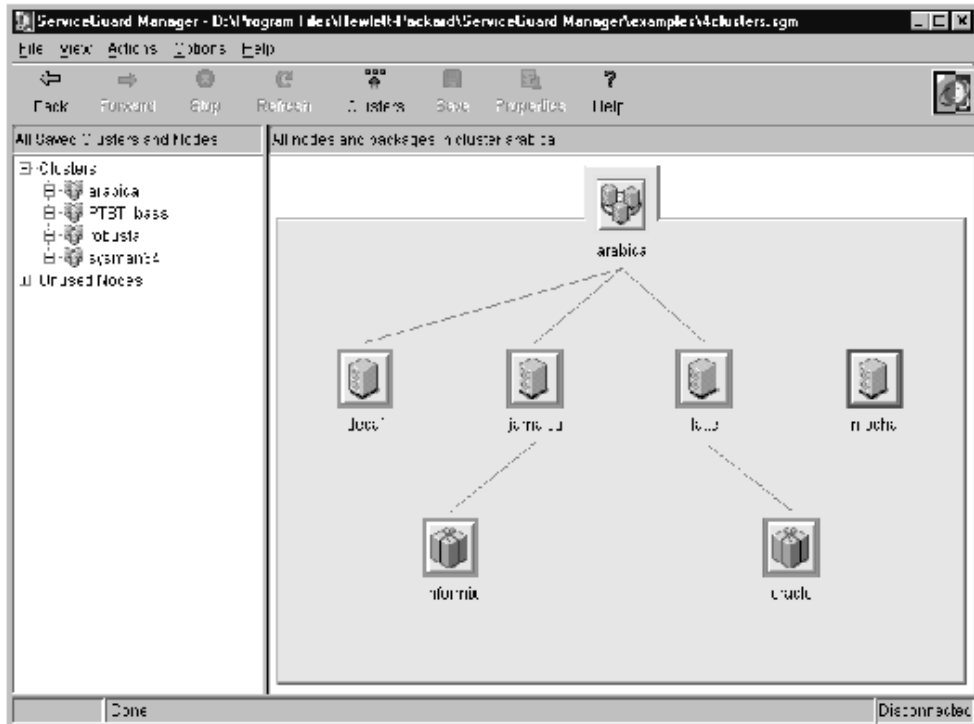
**Figure 7-5** Map and Cluster List Screen





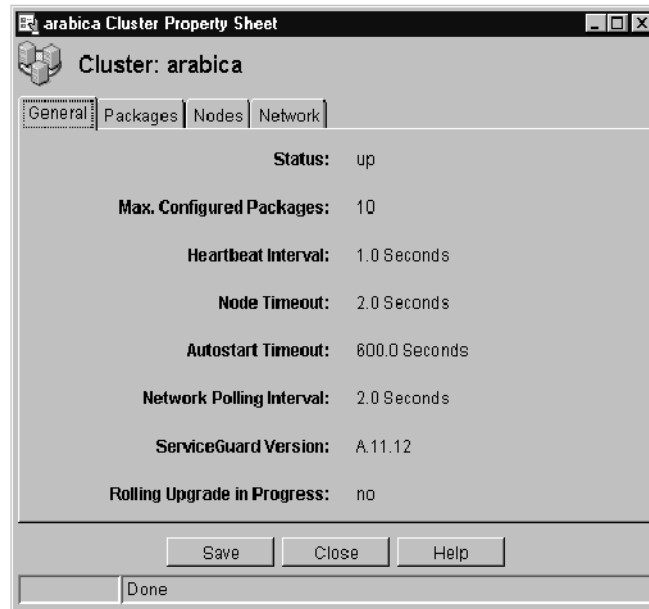
To display the objects within a specific cluster, double click the cluster icon, which will display a screen like the one in Figure 7-6.

**Figure 7-6 Cluster Details Screen**



For a display of an object's property sheet, click on the object, then click on "Properties" in the tool bar at the top of the screen. A cluster property sheet is shown in Figure 7-7.

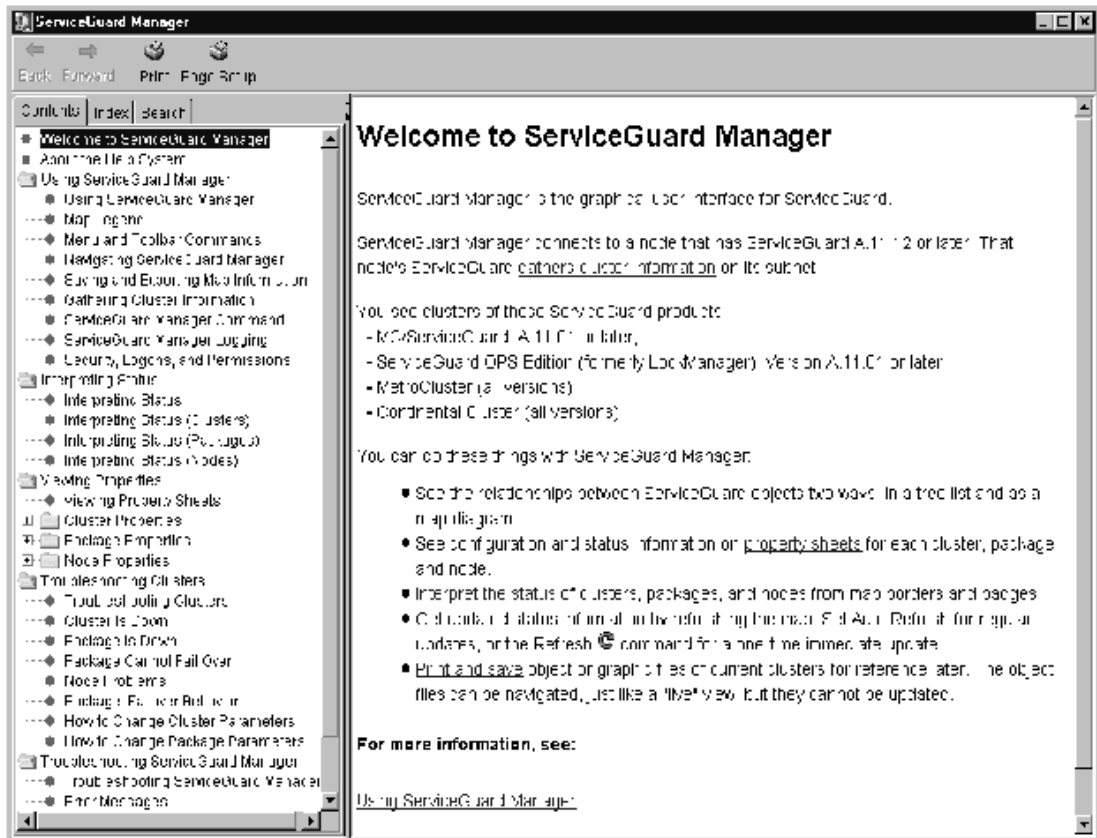
**Figure 7-7**      **Cluster Property Sheet Screen**



## Obtaining Help

To obtain online help, click on the “Help” menu item at the top of the screen. The top level help screen is shown in Figure 7-8.

**Figure 7-8** Help Screen



The following help topics under “Using Serviceguard Manager” are especially valuable for new users of the interface:

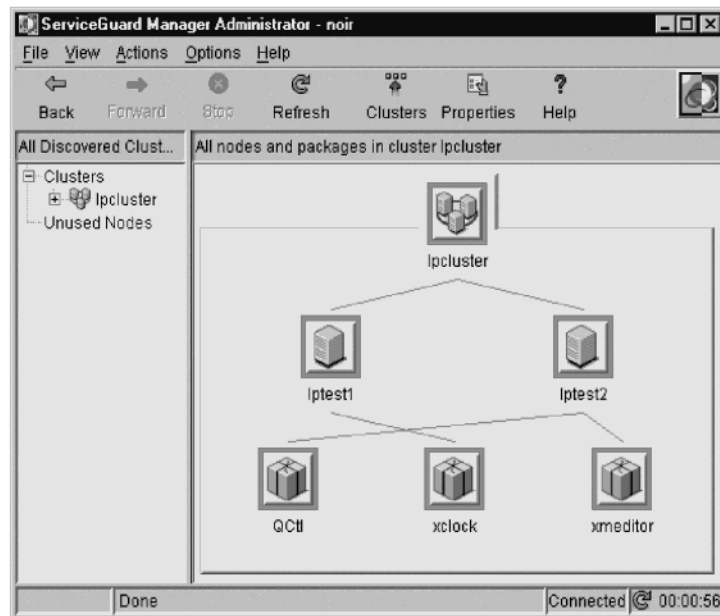
- “Menu and Toolbar Commands”
- “Navigating Serviceguard Manager”
- “Map Legend”

## Managing Cluster Objects

You can use Serviceguard Manager to manipulate clusters, nodes, and packages. It is possible to start and halt the cluster (including all packages), bring individual nodes in and out of cluster operation, start or stop individual packages, and move packages from one node to another. To perform management operations, select the cluster object, then display the Actions menu, or else right-click the cluster object to display a menu that includes possible actions. If a specific action is not appropriate for a cluster object, it will be greyed out.

An example of cluster management screen is shown in Figure 7-9. The cluster icon is right-clicked to display a menu of actions. If you click on **Start the Cluster**, Serviceguard Manager then issues the commands to start up the cluster.

**Figure 7-9** Cluster Map Showing Action Menu



The GUI displays a progress box that shows the Serviceguard commands that are being run to start the cluster (Figure 7-10).

**Figure 7-10**      **Cluster Startup Commands in Serviceguard Manager**

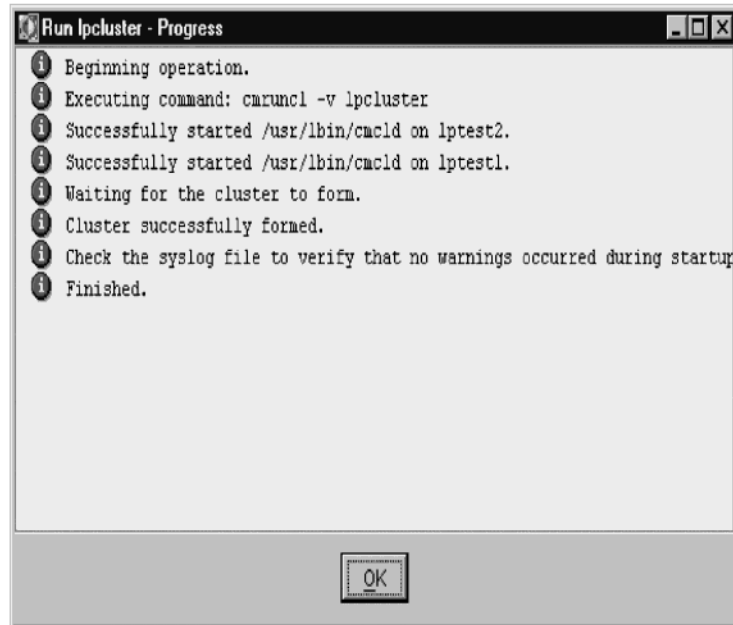
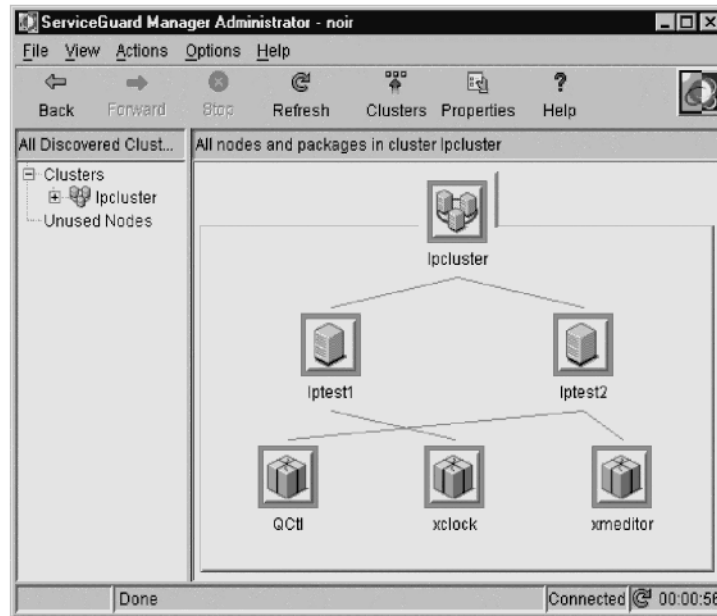


Figure 7-11 shows the outcome—a running cluster.

**Figure 7-11**      **Running Cluster**



You can do the following cluster administration tasks with Serviceguard Manager by selecting the object and activating the action menu:

- Start the cluster
- Halt the cluster
- Add a previously configured node to the cluster
- Halt a cluster node
- Start a package
- Halt a package
- Move a package from one node to another
- Reset switching parameters for packages and nodes

---

**CAUTION**

In order to carry out administrative operations, the user of a Serviceguard Manager client must log into the Object Manager system as *root*, using the root password. This effectively grants to the client the ability to manage all clusters that are serviced by that Object Manager. If it is not necessary for some users to administer the clusters, you can have them log in as ordinary users on the Object Manager system. In this case, they will be able to monitor the clusters but not administer them.

---

## Viewing Status of Monitored Disks

You can view the status of monitored disks by accessing the monitor server through the following URL:

**http://hostname:3542**

Supported web browsers include Internet Explorer 6.0 on Windows and Mozilla 1.0 on Linux. This will display a report similar to the following for the objects that are configured:

**Figure 7-12**      **Display of Monitored Objects**

Name	Current Status	Previous Status	Monitoring Node	Monitoring Start Time	Agent Timeout (seconds)	Agent Running Time (seconds)
Resource1	Good (2002-03-11 15:13:55)	Initiating (2002-03-11 15:13:55)	192.168.0.10.10000	2002-03-11 15:13:55	30.000000	0.000000
Resource2	Good (2002-03-11 15:13:55)	Initiating (2002-03-11 15:13:55)	192.168.0.10.10000	2002-03-11 15:13:55	30.000000	0.000000
Resource3	Good (2002-03-11 15:13:55)	Initiating (2002-03-11 15:13:55)	192.168.0.10.10000	2002-03-11 15:13:55	30.000000	0.000000

pkg15457\_2

Name	Current Status	Previous Status	Monitoring Node	Monitoring Start Time	Agent Timeout (seconds)	Agent Running Time (seconds)
Resource4	Good (2002-03-11 15:13:55)	Initiating (2002-03-11 15:13:55)	192.168.0.10.10000	2002-03-11 15:13:55	30.000000	0.000000

Copyright © 2002 by Oracle Corporation

Alternatively, use the `cmviewres` command to display the current status of system resources monitored by `cmresmond` on the local node. If no resources are specified, `cmviewres` displays the status of all system resources currently monitored by `cmresmond`.

```
# cmviewres <resource_name>
```



The following is a sample output, with resource `/dev/sdb1` specified is:

```
# cmviewres /dev/sdb1

Resource: /dev/sdb1
Category: pkg23837_1
Status: down
Timeout (seconds): 60.00
Polling Interval (seconds): 60.00
```

## Managing the Cluster and Nodes

Managing the cluster involves the following tasks:

- Starting the Cluster When All Nodes are Down
- Adding Previously Configured Nodes to a Running Cluster
- Removing Nodes from Operation in a Running Cluster
- Halting the Entire Cluster
- Reconfiguring a Halted Cluster

Starting the cluster means running the cluster daemon on one or more of the nodes in a cluster. You use different Serviceguard commands to start the cluster depending on whether all nodes are currently down (that is, no cluster daemons are running), or whether you are starting the cluster daemon on an individual node.

Note the distinction that is made in this chapter between adding an already configured node to the cluster and adding a new node to the cluster configuration. An already configured node is one that is already entered in the cluster configuration file; a new node is added to the cluster by modifying the cluster configuration file.

---

### NOTE

Manually starting or halting the cluster or individual nodes does not require access to the quorum server, if one is configured. The quorum server is only used when tie-breaking is needed following a cluster partition.

---

## Starting the Cluster When all Nodes are Down

### Using Serviceguard Manager to Start the Cluster

Select the cluster icon, then right-click to display the action menu. Select “Start the Cluster.” The progress window shows messages as the action takes place. This will include messages for starting each node and package. Click OK on the progress window when the operation is complete.

### Using HP Serviceguard Commands to Start the Cluster

Use the `cmrunc1` command to start the cluster when all cluster nodes are down. Particular command options can be used to start the cluster under specific circumstances.

The `-v` option to display the greatest number of messages. The following starts all nodes configured in the cluster without a connectivity check:

```
# cmrunc1 -v
```

The `-w` option causes `cmrunc1` to perform a full check of LAN connectivity among all the nodes of the cluster. Omitting this option will allow the cluster to start more quickly but will not test connectivity. The following starts all nodes configured in the cluster with a connectivity check:

```
# cmrunc1 -v -w
```

The `-n` option specifies a particular group of nodes. Without this option, all nodes will be started. The following example starts up the locally configured cluster only on `ftsys9` and `ftsys10`. (This form of the command should only be used when you are sure that the cluster is not already running on any node.)

```
# cmrunc1 -v -n ftsys9 -n ftsys10
```

---

#### CAUTION

HP Serviceguard cannot guarantee data integrity if you try to start a cluster with the `cmrunc1 -n` command while a subset of the cluster's nodes are already running a cluster. If the network connection is down between nodes, using `cmrunc1 -n` might result in a second cluster forming, and this second cluster might start up the same applications that are already running on the other cluster. The result could be two applications overwriting each other's data on the disks.

---

## Adding Previously Configured Nodes to a Running Cluster

You can use Serviceguard Manager or HP Serviceguard commands to bring a configured node up within a running cluster.

### Using Serviceguard Manager to Add a Node to the Cluster

Select the node icon, then right-click to display the action menu. Select “Start the Node.” The progress window shows messages as the action takes place. This will include starting any packages that are eligible to run on the node. Click OK on the progress window when the operation is complete.

### Using HP Serviceguard Commands to Add Previously Configured Nodes to a Running Cluster

Use the `cmrunnode` command to add one or more nodes to an already running cluster. Any node you add must already be a part of the cluster configuration. The following example adds node `ftsys8` to the cluster that was just started with only nodes `ftsys9` and `ftsys10`:

```
# cmrunnode -v ftsys8
```

Since the node's cluster is already running, the node joins the cluster and packages may be started. If the node does not find its cluster running, or the node is not part of the cluster configuration, the command fails.

## Removing Nodes from Operation in a Running Cluster

You can use Serviceguard Manager or HP Serviceguard commands to remove nodes from operation in a cluster. This operation removes the node from cluster operation by halting the cluster daemon, but it does not modify the cluster configuration. To remove a node from the cluster configuration permanently, you must recreate the cluster configuration file. See the next section.

### Using Serviceguard Manager to Remove a Node from the Cluster

Select the node icon, then right-click to display the action menu. Select “Halt the Node.” The progress window shows messages as the action takes place. This will include moving any packages on the node to adoptive nodes, if appropriate. Click OK on the progress window when the operation is complete.

### Using HP Serviceguard Commands to Remove Nodes from Operation

Use the `cmhaltnode` command to halt one or more nodes in a cluster. The cluster daemon on the specified node stops, and the node is removed from active participation in the cluster.

To halt a node with a running package, use the `-f` option. If a package was running that can be switched to an adoptive node, the switch takes place and the package starts on the adoptive node. For example, the following command causes the Serviceguard daemon running on node *ftsys9* in the sample configuration to halt and the package running on *ftsys9* to move to *ftsys10*:

```
# cmhaltnode -f -v ftsys9
```

This halts any packages running on the node *ftsys9* by executing the halt instructions in each package's control script. *ftsys9* is halted and the packages start on the adoptive node, *ftsys10*.

The use of `cmhaltnode` is a convenient way of bringing a node down for system maintenance while keeping its packages available on other nodes. After maintenance, the package can be returned to its primary node. See “Moving a Package,” below.

To return a node to the cluster, use `cmrunnode`.

---

**NOTE**

It is recommended to run `cmhaltnode` prior to running the HP-UX `shutdown` command, especially for cases where a packaged application might have trouble during shutdown and not halt cleanly.

---

## Halting the Entire Cluster

You can use Serviceguard Manager or HP Serviceguard commands to halt a running cluster.

### Using Serviceguard Manager to Halt the Cluster

Select the cluster, then right-click to display the action menu. Select “Halt the Cluster.” The progress window shows messages as the action takes place. This will include messages for halting each package and node. Click OK on the progress window when the operation is complete.

### Using HP Serviceguard Commands to Halt a Cluster

The `cmhaltcl` command can be used to halt the entire cluster. This command causes all nodes in a configured cluster to halt their HP Serviceguard daemons. You can use the `-f` option to force the cluster to halt even when packages are running. This command can be issued from any running node. Example:

```
# cmhaltcl -f -v
```

This halts all nodes that are configured in the cluster.

## Reconfiguring a Halted Cluster

You can also make a permanent change in cluster configuration when the cluster is halted. This procedure *must* be used for changes to the quorum server configuration, changes in timing parameters, and changes to the Maximum Number of Packages parameter, but it can be used for any other cluster configuration changes as well.

Use the following steps:

1. Halt the cluster on all nodes.
2. On one node, reconfigure the cluster as described in the chapter “Building an HA Cluster Configuration.” You can use the `cmgetconf` command to generate an ASCII file, which you then edit.
3. Make sure that all nodes listed in the cluster configuration file are powered up and accessible. Use `cmapplyconf` to copy the binary cluster configuration file to all nodes. This file overwrites any previous version of the binary cluster configuration file.
4. Use the `cmruncl` command to start the cluster on all nodes or on a subset of nodes, as desired.

### Changing `MAX_CONFIGURED_PACKAGES`

Use the `cmgetconf` command to obtain a current copy of the cluster's existing configuration. Example:

```
# cmgetconf -C clconfig.ascii
```

Edit the `clconfig.ascii` file to include the desired value for `MAX_CONFIGURED_PACKAGES`. Then use the `cmcheckconf` command to verify the new configuration. Use the `cmapplyconf` command to apply the changes to the configuration and send the new configuration file to all cluster nodes.

## Automatically Restarting the Cluster

You can configure your cluster to automatically restart after an event, such as a long-term power failure, which brought down all nodes in the cluster. This is done by setting `AUTOSTART_CMCLD` to 1 in the `$SGAUTOSTART` file.

---

## Reconfiguring a Running Cluster

You can add new nodes to the cluster configuration or delete nodes from the cluster configuration while the cluster is up and running. Note the following, however:

- You cannot remove an active node from the cluster. You must halt the node first.
- You cannot change cluster timing parameters.
- The only configuration change allowed while a node is unreachable (for example, completely disconnected from the network) is to delete the unreachable node from the cluster configuration. If there are also packages that depend upon that node, the package configuration must also be modified to delete the node. This all must be done in one configuration request (`cmapplyconf` command).
- The access control list for the cluster can be changed while the cluster is running.

Changes to the package configuration are described in a later section.

**Table 7-2**

**Types of Changes to Permanent Cluster Configuration**

Change to the Cluster Configuration	Required Cluster State
Add a new node	All cluster nodes must be running.
Delete a node	A node can be deleted even though it is unavailable or unreachable.
Change Maximum Configured Packages	Cluster must not be running.
Change Timing Parameters	Cluster must not be running.
Change Lock LUN Configuration	Cluster must not be running.
Change Quorum Server Configuration	Cluster must not be running.



**Table 7-2**      **Types of Changes to Permanent Cluster Configuration**

Change to the Cluster Configuration	Required Cluster State
Change IP addresses for heartbeats or monitored subnets	Cluster must not be running.

The following sections describe how to perform dynamic reconfiguration tasks.

## Adding Nodes to the Configuration While the Cluster is Running

Use the following procedure to add a node. For this example, nodes *ftsys8* and *ftsys9* are already configured in a running cluster named *cluster1*, and you are adding node *ftsys10*.

1. Use the following command to store a current copy of the existing cluster configuration in a temporary file:

```
# cmgetconf -C temp.ascii
```

2. Specify a new set of nodes to be configured and generate a template of the new configuration:

```
# cmquerycl -C clconfig.ascii -c cluster1 \
-n ftsys8 -n ftsys9 -n ftsys10
```

3. Edit the file `clconfig.ascii` to check the information about the new node.

4. Verify the new configuration:

```
# cmcheckconf -C clconfig.ascii
```

5. Apply the changes to the configuration and send the new binary configuration file to all cluster nodes:

```
# cmapplyconf -C clconfig.ascii
```

Use `cmrunnode` to start the new node, and, if desired, set the `AUTOSTART_CMCLD` parameter to 1 in the `$SGAUTOSTART` file to enable the new node to join the cluster automatically each time it reboots.

## Deleting Nodes from the Configuration While the Cluster is Running

Use the following procedure to delete a node. For this example, nodes *ftsys8*, *ftsys9* and *ftsys10* are already configured in a running cluster named *cluster1*, and you are deleting node *ftsys10*.

1. Halt the node, which you are planning to remove by using the following command:

```
# cmhaltnode -f ftsys10
```

2. Use the following command to store a current copy of the existing cluster configuration in a temporary file:

```
# cmgetconf -c cluster1 temp.ascii
```

3. Specify the new set of nodes to be configured (omitting *ftsys10*) and generate a template of the new configuration:

```
# cmquerycl -C clconfig.ascii -c cluster1 -n ftsys8 \
-n ftsys9
```

4. Edit the file *clconfig.ascii* to check the information about the nodes that remain in the cluster.

5. Verify the new configuration:

```
# cmcheckconf -C clconfig.ascii
```

6. Apply the changes to the configuration and send the new binary configuration file to all cluster nodes:

```
# cmapplyconf -C clconfig.ascii
```

Use *cmrunnode* to start the new node, and, if desired, set the *AUTOSTART\_CMCLD* parameter to 1 in the *\$SGAUTOSTART* file to enable the new node to join the cluster automatically each time it reboots.

## Managing Packages and Services

Managing packages and services involves the following tasks:

- Starting a Package
- Halting a Package
- Moving a Package
- Reconfiguring a Package on a Halted Cluster

### Starting a Package

Ordinarily, a package configured as part of the cluster will start up on its primary node when the cluster starts up. You may need to start a package manually after it has been halted manually. You can do this either in Serviceguard Manager or with HP Serviceguard commands.

#### Using Serviceguard Manager to Start a Package

Select the package you wish to start, and right-click to display the action list. You can start the package either on its default configured node, or on any node in the package node list. Select “Start Package on Configured Node” or “Start Package on Specified Node.” In the latter case, you will see a select list of running eligible nodes from which you can choose the node on which the package should start.

The progress window shows messages as the action takes place. This will include a messages for starting the package.

The cluster must be running in order to start a package.

#### Using HP Serviceguard Commands to Start a Package

Use the `cmrunpkg` command to run the package on a particular node, then use the `cmmodpkg` command to enable switching for the package. Example:

```
# cmrunpkg -n ftsys9 pkg1
# cmmodpkg -e pkg1
```

This starts up the package on *ftsys9*, then enables package switching. This sequence is necessary when a package has previously been halted on some node, since halting the package disables switching.

## **Halting a Package**

You halt an HP Serviceguard package when you wish to bring the package out of use but wish the node to continue in operation. You can halt a package using Serviceguard Manager or HP Serviceguard commands. Halting a package has a different effect than halting the node. When you halt the node, its packages may switch to adoptive nodes (assuming that switching is enabled for them); when you halt the package, it is disabled from switching to another node, and must be restarted manually on another node or on the same node.

### **Using Serviceguard Manager to Halt a Package**

Select the package you wish to halt, and right-click to display the action list. Select “Halt the package.” The package must be running.

The progress window shows messages as the action takes place. This will include a message for halting the package.

### **Using HP Serviceguard Commands to Halt a Package**

Use the `cmhaltpkg` command to halt a package, as follows:

```
# cmhaltpkg pkg1
```

This halts `pkg1` and disables it from switching to another node.

## Moving a Package

You can use Serviceguard Manager or HP Serviceguard commands to move a package from one node to another.

### Using Serviceguard Manager to Move a Package

Select the icon of the package you wish to halt, and right-click to display the action list. Select “Move package to node.” The package must be running.

You will see a list of possible destinations. Click on the node where you want the package to run.

The progress window shows messages as the action takes place. This will include a message for halting the package and another for starting it on the destination node.

### Using HP Serviceguard Commands to Move a Running Package

Before you move the package, halt it on its original node using the `cmhaltpkg` command. This action not only halts the package, but also disables switching the package back to the node on which it halts.

After you have moved the package you must restart it and enable switching. You can do this by issuing the `cmrunpkg` command followed by `cmmodpkg -e package_name`. `cmmodpkg` can be used with the `-n` option to enable a package to run on a node if the package has been disabled from running on that node due to some sort of error. If no node is specified, the node the command is run on is the implied node.

Example:

```
# cmhaltpkg pkg1
# cmrunpkg -n ftsys10 pkg1
# cmmodpkg -e pkg1
```

## **Reconfiguring a Package on a Halted Cluster**

You can also make permanent changes in package configuration while the cluster is not running. Use the following steps:

- On one node, reconfigure the package as described earlier in chapter 6. You can do this by editing the package ASCII file.
- Edit the package control script. Any changes in service names will also require changes in the package configuration file.
- Use the `cmapplyconf` command to copy the binary cluster configuration file to all nodes. Use the `-P` option, specifying the package to be changed; do not use the `-C` option. This file overwrites any previous version of the binary cluster configuration file.
- Copy the modified control script to all nodes that can run the package.
- Use the `cmruncl` command to start the cluster on all nodes or on a subset of nodes, as desired. The package will start up as nodes come online.

## Reconfiguring a Package on a Running Cluster

You can reconfigure a package while the cluster is running, and in some cases you can reconfigure the package while the package itself is running. Only certain changes may be made while the package is running.

To modify the package, use the following procedure (*pkg1* is used as an example):

1. Halt the package if necessary:

```
# cmhaltpkg pkg1
```

See Table 7-3 to determine whether this step is needed.

2. If it is not already available, you can obtain a copy of the package's ASCII configuration file by using the `cmgetconf` command, specifying the package name.

```
# cmgetconf -P pkg1.ascii
```

3. Edit the ASCII package configuration file.

4. Verify your changes as follows:

```
# cmcheckconf -v -P pkg1.ascii
```

5. Distribute your changes to all nodes:

```
# cmapplyconf -v -P pkg1.ascii
```

6. Copy the package control script to all nodes that can run the package.

## Adding a Package to a Running Cluster

You can create a new package and add it to the cluster configuration while the cluster is up and while other packages are running. The number of packages you can add is subject to the value of *Maximum Configured Packages* as defined in the cluster configuration file.

To create the package, follow the steps given in the chapter “Configuring Packages and Services” with the following difference: *do not* specify the cluster ASCII file when verifying and distributing the configuration with the `cmapplyconf` command. For example, to verify the configuration of newly created *pkg1* on a running cluster:

```
# cmcheckconf -P $SGCONF/pkg1/pkg1conf.ascii
```

Use a command like the following to distribute the new package configuration to all nodes in the cluster:

```
# cmapplyconf -P $SGCONF/conf/pkg1/pkg1conf.ascii
```

Remember to copy the control script to the `$SGCONF/pkg1` directory on all nodes that can run the package.

## Deleting a Package from a Running Cluster

You can delete a package from all cluster nodes by using the `cmdeleteconf` command. The command can only be executed when the package is not running; the cluster may be up. The command removes the package information from the binary configuration file on all the nodes in the cluster.

The following example halts package *mypkg* and removes the package configuration from the cluster:

```
# cmhaltpkg mypkg  
# cmdeleteconf -p mypkg
```

The command prompts for a verification before deleting the files unless you use the `-f` option. The directory `$SGCONF/mypkg` is not deleted by this command.



## Changing Package Switching Behavior

You can change package switching behavior either temporarily or permanently. To temporarily disable switching to other nodes for a running package, use the `cmmodpkg` command. For example, if *pkg1* is currently running, and you want to disable its ability to start up on another node, enter the following:

```
# cmmodpkg -d pkg1
```

This does not halt the package, but it will prevent the package from starting up elsewhere.

To permanently disable switching so that the next time the cluster restarts, the change you made in package switching is still in effect, you must change the `AUTO_RUN` flag in the package configuration file, then re-apply the configuration. (Any change made this way will take effect the next time the cluster is restarted.)

See the previous section “Reconfiguring a Package on a Running Cluster” for detailed instructions on reconfiguration.

## Resetting the Service Restart Counter

The service restart counter is the number of times a package service has been automatically restarted. This value is used to determine when the package service has exceeded its maximum number of allowable automatic restarts.

---

### NOTE

The maximum number of allowable restarts for a given service is set in the package control script parameter `SERVICE_RESTART[]`. This parameter is not the same as the restart counter, which is maintained separately by the package manager.

---

When a package service successfully restarts after several attempts, the package manager does not automatically reset the restart count. However, you may choose to reset the counter online using the `cmmodpkg -R -s` command, thus enabling the service in future restart situations to have the full number of restart attempts up to the configured `SERVICE_RESTART` count. Example:

```
# cmmodpkg -R -s myservice pkg1
```

The current value of the restart counter may be seen in the output of the `cmviewcl -v` command.

## Allowable Package States During Reconfiguration

All nodes in the cluster must be powered up and accessible when making configuration changes.

Refer to Table 7-3 to determine whether or not the package may be running while you implement a particular kind of change. Note that for all of the following cases the cluster may be running, and also packages other than the one being reconfigured may be running.

**Table 7-3**      **Types of Changes to Packages**

<b>Change to the Package</b>	<b>Required Package State</b>
Add a new package	Other packages may be in any state.
Delete a package	Package must not be running.
Remove a node from a package	Package must not be running on the node to be removed. The package can be running on another node in the cluster.
Add a service	Package must not be running.
Remove a service	Package must not be running.
Add a subnet	Package must not be running. Subnet must already be configured into the cluster.
Remove a subnet	Package must not be running.
Add a volume group	Volume group may be configured into the package while the cluster is running. The package may be in any state, because the change is made in the control script. However, the package must be halted and restarted for the change to have an effect.
Remove a volume group	Package must not be running.

**Table 7-3**                      **Types of Changes to Packages (Continued)**

<b>Change to the Package</b>	<b>Required Package State</b>
Change run script contents	It is recommended that the package be halted. If the run script for the package is modified while the package is running, timing may cause problems.
Change halt script contents	It is recommended that the package be halted. If the halt script for the package is modified while the package is running, timing may cause problems.
Script timeouts	Package may be either running or halted.
Service timeouts	Package must not be running.
Service failfast	Package must not be running.
Package Auto_Run	Package may be either running or halted.
Change the order of nodes where a package may run	Package may be either running or halted.
Change the Package Failover Policy	Package may be either running or halted.
Change the Package Failback Policy	Package may be either running or halted.

## **Responding to Cluster Events**

Serviceguard does not require much ongoing system administration intervention. As long as there are no failures, your cluster will be monitored and protected. In the event of a failure, those packages that you have designated to be transferred to another node will be transferred automatically. Your ongoing responsibility as the system administrator will be to monitor the cluster and determine if a transfer of package has occurred. If a transfer has occurred, you have to determine the cause and take corrective actions.

The typical corrective actions to take in the event of a transfer of package include:

- Determining when a transfer has occurred.
- Determining the cause of a transfer.
- Repairing any hardware failures.
- Correcting any software problems.
- Restarting nodes.
- Transferring packages back to their original nodes.
- Enabling package switching.

## Single-Node Operation

The number of nodes you will need for your Serviceguard cluster depends on the processing requirements of the applications you want to protect. In a multi-node cluster, you could have a situation where all but one node has failed, or where you have shut down all but one node, leaving your cluster in single-node operation. This remaining node will probably have applications running on it. As long as the Serviceguard daemon `cmclld` is active, other nodes can re-join the cluster at a later time.

If the Serviceguard daemon fails when in single-node operation, it will leave the single node up and your applications running. This is different from the loss of the Serviceguard daemon in a multi-node cluster, which halts the node with a TOC, and causes packages to be switched to adoptive nodes. It is not necessary to halt the single node in this scenario, since the application is still running, and no other node is currently available for package switching.

You should *not* try to restart Serviceguard, since data corruption might occur if another node were to attempt to start up a new instance of the application that is still running on the single node.

Instead of restarting the cluster, choose an appropriate time to shutdown and reboot the node, which will allow the applications to shut down and then permit Serviceguard to restart the cluster after rebooting.

## **Removing Serviceguard from a System**

If you wish to remove a node from Serviceguard use, use the `rpm -e` command to delete the software. Note the following:

- The cluster should not be running on the node from which you will be deleting Serviceguard.
- The node from which you are deleting Serviceguard should not be in the cluster configuration.
- If you are removing Serviceguard from more than one node, `rpm -e` should be issued on one node at a time.

# 8 Troubleshooting Your Cluster

This chapter describes how to verify cluster operation, how to review cluster status, how to add and replace hardware, and how to solve some typical cluster problems. Topics are as follows:

- Testing Cluster Operation
- Monitoring Hardware
- Replacing Disks
- Replacement of LAN Cards
- Replacing a Failed Quorum Server System
- Troubleshooting Approaches
- Solving Problems

## Testing Cluster Operation

Once you have configured your Serviceguard cluster, you should verify that the various components of the cluster behave correctly in case of a failure. In this section, the following procedures test that the cluster responds properly in the event of a package failure, a node failure, or a LAN failure.

---

### CAUTION

In testing the cluster in the following procedures, be aware that you are causing various components of the cluster to fail, so that you can determine that the cluster responds correctly to failure situations. As a result, the availability of nodes and applications may be disrupted.

---

## Testing the Package Manager

To test that the package manager is operating correctly, perform the following procedure for each package on the cluster:

1. Obtain the PID number of a service in the package by entering

```
# ps -ef | grep <service_cmd>
```

where *service\_cmd* is the executable specified in the package control script with the parameter `SERVICE_CMD`. The service selected must not have `SERVICE_RESTART` specified.

2. To kill the *service\_cmd* PID, enter

```
# kill PID
```

3. To view the package status, enter

```
# cmviewcl -v
```

The package should be running on the specified adoptive node.

4. Halt the package, then move it back to the primary node using the `cmhaltpkg`, `cmmodpkg`, and `cmrunpkg` commands:

```
# cmhaltpkg <PackageName>
```

```
# cmmodpkg -e <PrimaryNode> <PackageName>
```

```
# cmrunpkg -v <PackageName>
```



Depending on the specific databases you are running, perform the appropriate database recovery.

## Testing the Cluster Manager

To test that the cluster manager is operating correctly, perform the following steps for each node on the cluster:

1. Turn off the power to the node.
2. To observe the cluster reforming, enter the following command on some other configured node:

```
# cmviewcl -v
```

You should be able to observe that the powered down node is halted, and that its packages have been correctly switched to other nodes.

3. Turn on the power to the node.
4. To verify that the node is rejoining the cluster, enter the following command on any configured node:

```
# cmviewcl -v
```

The node should be recognized by the cluster, but its packages should *not* be running.

5. Move the packages back to the original node:

```
# cmhaltpkg <pkgname>
```

```
# cmmodpkg -e -n <originalnode>
```

```
# cmrunpkg <pkgname>
```

Depending on the specific databases you are running, perform the appropriate database recovery.

6. Repeat this procedure for all nodes in the cluster one at a time.

## Testing the Network Manager

To test that the Network Manager is operating correctly, do the following for each node in the cluster:

1. Identify the LAN cards on the node:

```
# ifconfig
```

and then

```
# cmviewcl -v
```

2. Detach the LAN connection from one card.

3. Use `cmviewcl` to verify that the network is still functioning through the other cards:

```
# cmviewcl -v
```

4. Reconnect the LAN to the original card, and verify its status:

```
# cmviewcl -v
```

## Monitoring Hardware

Good standard practice in handling a high availability system includes careful fault monitoring so as to prevent failures if possible or at least to react to them swiftly when they occur. Disks can be monitored using the Disk Monitor daemon, which is described in Chapter 5. In addition, the following should be monitored for errors or warnings of all kinds:

- CPUs
- Memory
- LAN cards
- Power sources
- All cables
- Disk interface cards

Some monitoring can be done through simple physical inspection, but for the most comprehensive monitoring, you should examine the system log file (`/var/log/messages`) periodically for reports on all configured HA devices. The presence of errors relating to a device will show the need for maintenance.

## Replacing Disks

The procedure for replacing a faulty disk mechanism depends on the type of disk configuration you are using. Refer to your Smart Array documentation for issues related to your Smart Array.

### Replacing a Faulty Mechanism in a Disk Array

You can replace a failed disk mechanism by simply removing it from the array and replacing it with a new mechanism of the same type. The resynchronization is handled by the array itself. There may be some impact on disk performance until the resynchronization is complete. For details on the process of hot plugging disk mechanisms, refer to your disk array documentation.

## Replacement of LAN Cards

If you need to replace a LAN card, use the following steps. It is not necessary to bring the cluster down to do this.

**Step 1.** Halt the node using the `cmhaltnode` command.

**Step 2.** Shut down the system:

```
shutdown -h
```

Then power off the system.

**Step 3.** Remove the defective LAN card.

**Step 4.** Install the new LAN card. The new card must be exactly the same card type, and it must be installed in the same slot as the card you removed.

**Step 5.** Power up the system.

**Step 6.** As the system comes up, the `kudzu` program on RedHat systems will detect and report the hardware changes. Accept the changes and add any information needed for the new LAN card. On SUSE systems, run `YAST2` after the system boots and make adjustments to the NIC setting of the new LAN card. If the old LAN card was part of a “bond”, the new LAN card needs to be made part of the bond. See “Implementing Channel Bonding (Red Hat)” on page 140 or “Implementing Channel Bonding (SLES9 and SLES10)” on page 144.

**Step 7.** If necessary, add the node back into the cluster using the `cmrunnode` command.

(You can omit this step if the node is configured to join the cluster automatically.)

Now Serviceguard will detect that the MAC address (LLA) of the card has changed from the value stored in the cluster binary configuration file, and it will notify the other nodes in the cluster of the new MAC address. The cluster will operate normally after this.

HP recommends that you update the new MAC address in the cluster binary configuration file by re-applying the cluster configuration. Use the following steps for online reconfiguration:

## Replacement of LAN Cards

1. Use the `cmgetconf` command to obtain a fresh ASCII configuration file, as follows:

```
# cmgetconf config.ascii
```

2. Use the `cmapplyconf` command to apply the configuration and copy the new binary file to all cluster nodes:

```
# cmapplyconf -C config.ascii
```

This procedure updates the binary file with the new MAC address and thus avoids data inconsistency between the outputs of the `cmviewconf` and `ifconfig` commands.

## Replacing a Failed Quorum Server System

When a quorum server fails or becomes unavailable to the clusters it is providing quorum services for, this will not cause a failure on any cluster. However, the loss of the quorum server does increase the vulnerability of the clusters in case there is an additional failure. Use the following procedure to replace a defective quorum server system. If you use this procedure, you do not need to change the configuration of any cluster nodes.

1. Remove the old quorum server system from the network.
2. Set up the new system and configure it with the old quorum server's IP address and hostname.
3. Install and configure the quorum server software on the new system. Be sure to include in the new QS authorization file (for example, `/usr/local/qs/conf/qs_authfile`) on all of the nodes that were configured for the old quorum server. Refer to the `qs(1)` man page for details about configuring the QS authorization file.

---

### NOTE

The quorum server reads the authorization file at startup. Whenever you modify the file `qs_authfile`, run the following command to force a re-read of the file. For example on a Red Hat distribution:

```
# /usr/local/qs/bin/qs -update
```

On a SuSE distribution:

```
# /opt/qs/bin/qs -update
```

- 
4. Start the quorum server as follows:
    - Edit the `/etc/inittab` file to add the quorum server entries, as shown in “Running the Quorum Server” on page 135 above.
    - Use the `init q` command to run the quorum server.

Or

- Create a package in another cluster for the Quorum Server as shown in “Creating a Package for the Quorum Server” on page 136 above.

Refer to the `qs(1)` man page for more details.

5. All nodes in all clusters that were using the old quorum server will connect to the new quorum server. Use the `cmviewcl -v` command from any cluster that is using the quorum server to verify that the nodes in that cluster have connected to the QS.
6. The quorum server log file on the new quorum server will show a log message like the following for each cluster that uses the quorum server:

```
Request for lock /sg/<ClusterName> succeeded. New lock
owners: N1, N2
```

7. To check that the quorum server has been correctly configured and to verify the connectivity of a node to the quorum server, you can execute the following command from your cluster nodes as follows:

```
# cmquerycl -q <QSHostName> -n <Node1> -n <Node2> ...
```

The command will output an error message if the specified nodes cannot communicate with the quorum server.

---

**NOTE**

While the old quorum server is down and the new one is being set up:

- The `cmquerycl`, `cmcheckconf` and `cmapplyconf` commands will not work
- The `cmruncl`, `cmhaltcl`, `cmrunnode`, and `cmhaltnode` commands will work
- If there is a node or network failure that creates a 50-50 membership split, the quorum server will not be available as a tie-breaker, and the cluster will fail.

---

**CAUTION**

**Make sure that the old system does not re-join the network with the old IP address.**

---



## Troubleshooting Approaches

The following sections offer a few suggestions for troubleshooting by reviewing the state of the running system and by examining cluster status data, log files, and configuration files. Topics include:

- Reviewing Package IP Addresses
- Reviewing the System Log File
- Reviewing Configuration Files
- Reviewing the Package Control Script
- Using `cmquerycl` and `cmcheckconf`
- Using `cmscancl` and `cmviewcl`
- Reviewing the LAN Configuration

---

### NOTE

The use of Serviceguard Manager is recommended for observing the current status of a cluster and viewing the properties of cluster objects. See “Using Serviceguard Manager” in Chapter 7 for information about running Serviceguard Manager.

---

## Reviewing Package IP Addresses

The `ifconfig` command can be used to examine the LAN configuration. The command, if executed on *ftsys9* after the halting of node *ftsys10*, shows that the package IP addresses are assigned to `eth1:1` and `eth1:2` along with the heartbeat IP address on `eth1`.

```
eth0      Link encap:Ethernet  HWaddr 00:01:02:77:82:75
          inet addr:15.13.169.106 Bcast:15.13.175.255 Mask:255.255.248.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:70826196 errors:0 dropped:0 overruns:1 frame:0
          TX packets:5741486 errors:1 dropped:0 overruns:1 carrier:896
          collisions:26706 txqueuelen:100
          Interrupt:9 Base address:0xdc00

eth1      Link encap:Ethernet  HWaddr 00:50:DA:64:8A:7C
          inet addr:192.168.1.106 Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2337841 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1171966 errors:0 dropped:0 overruns:0 carrier:0
          collisions:6 txqueuelen:100
          Interrupt:9 Base address:0xda00

eth1:1    Link encap:Ethernet  HWaddr 00:50:DA:64:8A:7C
          inet addr:192.168.1.200 Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:9 Base address:0xda00

eth1:2    Link encap:Ethernet  HWaddr 00:50:DA:64:8A:7C
          inet addr:192.168.1.201 Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:9 Base address:0xda00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Bcast:192.168.1.255 Mask:255.255.255.0
          UP LOOPBACK RUNNING MULTICAST  MTU:3924 Metric:1
          RX packets:2562940 errors:0 dropped:0 overruns:1 frame:0
          TX packets:2562940 errors:1 dropped:0 overruns:1 carrier:896
          collisions:0 txqueuelen:0
```

## Reviewing the System Log File

Messages from the Cluster Manager and Package Manager are written to the system log file. The default location of the log file may vary according to Linux distribution; the Red Hat default is `/var/log/messages`. You can use a text editor, such as `vi`, or the `more` command to view the log file for historical information on your cluster.

This log provides information on the following:

- Commands executed and their outcome.
- Major cluster events which may, or may not, be errors.
- Cluster status information.

---

### NOTE

Many other products running on Linux in addition to Serviceguard use the `syslog` file to save messages. Refer to your Linux documentation for additional information on using the system log.

---

## Sample System Log Entries

The following entries from the syslog file show a package that failed to run due to a problem in the `pkg5_run` script. You would look at the `pkg5_run.log` for details. Refer to “Understanding the Location of Serviceguard Files” on page 122 for references to `$SGCONF`.

```
Dec 14 14:33:48 star04 cmcld[2048]: Starting cluster management protocols.
Dec 14 14:33:48 star04 cmcld[2048]: Attempting to form a new cluster
Dec 14 14:33:53 star04 cmcld[2048]: 3 nodes have formed a new cluster
Dec 14 14:33:53 star04 cmcld[2048]: The new active cluster membership is:
    star04(id=1) , star05(id=2), star06(id=3)
Dec 14 17:33:53 star04 cmlvmd[2049]: Clvmd initialized successfully.
Dec 14 14:34:44 star04 CM-CMD[2054]: cmrunpkg -v pkg5
Dec 14 14:34:44 star04 cmcld[2048]: Request from node star04 to start
    package pkg5 on node star04.
Dec 14 14:34:44 star04 cmcld[2048]: Executing '/usr/local/cmcluster/conf/pkg5/pk
g5_run
start' for package pkg5.
Dec 14 14:34:45 star04 LVM[2066]: vgchange -a n /dev/vg02
Dec 14 14:34:45 star04 cmcld[2048]: Package pkg5 run script exited with
    NO_RESTART.
Dec 14 14:34:45 star04 cmcld[2048]: Examine the file
    /etc/cmcluster/pkg5/pkg5_run.log for more details.
```

The following is an example of a successful package starting:

```
Dec 14 14:39:27 star04 CM-CMD[2096]: cmruncl
Dec 14 14:39:27 star04 cmcld[2098]: Starting cluster management protocols.
Dec 14 14:39:27 star04 cmcld[2098]: Attempting to form a new cluster
Dec 14 14:39:27 star04 cmclconfd[2097]: Command execution message
Dec 14 14:39:33 star04 cmcld[2098]: 3 nodes have formed a new cluster
Dec 14 14:39:33 star04 cmcld[2098]: The new active cluster membership is:
    star04(id=1), star05(id=2), star06(id=3)
Dec 14 17:39:33 star04 cmlvmd[2099]: Clvmd initialized successfully.
Dec 14 14:39:34 star04 cmcld[2098]: Executing '/usr/local/cmcluster/conf/pkg4/pk
g4_run
start' for package pkg4.
Dec 14 14:39:34 star04 LVM[2107]: vgchange /dev/vg01
Dec 14 14:39:35 star04 CM-pkg4[2124]: cmmmodnet -a -i 15.13.168.0 15.13.168.4
Dec 14 14:39:36 star04 CM-pkg4[2127]: cmrunserv Service4 /vg01/MyPing 127.0.0.1
    >>/dev/null
Dec 14 14:39:36 star04 cmcld[2098]: Started package pkg4 on node star04.
```

## Reviewing Object Manager Log Files

The Serviceguard Object Manager daemon `cmomd` logs messages to the file `/usr/local/cmom/cmomd.log` on Red Hat and `/var/log/cmmomcmomd.log` on SuSE. You can review these messages using the `cmreadlog` command, for example:

```
# /usr/local/cmom/bin/cmreadlog /usr/local/cmom/log/cmomd.log
```

Messages from `cmomd` include information about the processes that request data from the Object Manager, including type of data, timestamp, etc. An example of a client that requests data from Object Manager is Serviceguard Manager.

## Reviewing Configuration Files

Review the following ASCII configuration files:

- Cluster configuration file.
- Package configuration files.

Ensure that the files are complete and correct according to your configuration planning worksheets.

## Reviewing the Package Control Script

Ensure that the package control script is found on all nodes where the package can run and that the file is identical on all nodes. Ensure that the script is executable on all nodes. Ensure that the name of the control script appears in the package configuration file, and ensure that all services named in the package configuration file also appear in the package control script.

Information about the starting and halting of each package is found in the package's control script log. This log provides the history of the operation of the package control script. It is found in the same directory as the control script itself. This log documents all package run and halt activities. If you have written a separate run and halt script for a package, each script will have its own log. Refer to "Understanding the Location of Serviceguard Files" on page 122 for references to `$SGCONF`.

## Using the `cmquerycl` and `cmcheckconf` Commands

In addition, `cmquerycl` and `cmcheckconf` can be used to troubleshoot your cluster just as they were used to verify its configuration. The following example shows the commands used to verify the existing cluster configuration on *ftsys9* and *ftsys10*:

```
# cmquerycl -v -C $SGCONF/verify.ascii -n ftsys9 -n ftsys10
# cmcheckconf -v -C $SGCONF/verify.ascii
```

The `cmcheckconf` command checks:

- The network addresses and connections.
- Quorum Server connectivity, if a quorum server is configured.
- Lock LUN connectivity, if a lock LUN is used.
- The validity of configuration parameters of the cluster and packages for:
  - The uniqueness of names.
  - The existence and permission of scripts.

It doesn't check:

- The correct setup of the power circuits.
- The correctness of the package configuration script.

## Reviewing the LAN Configuration

The following networking commands can be used to diagnose problems:

- `ifconfig` can be used to examine the LAN configuration. This command lists all IP addresses assigned to each LAN interface card.
- `arp -a` can be used to check the arp tables.
- `cmscancl` can be used to test IP-level connectivity between network interfaces in the cluster.
- `cmviewcl -v` shows the status of primary LANs.

Use these commands on all nodes.

## Solving Problems

Problems with Serviceguard may be of several types. The following is a list of common categories of problem:

- Serviceguard Command Hangs.
- Cluster Re-formations.
- System Administration Errors.
- Package Control Script Hangs.
- Package Movement Errors.
- Node and Network Failures.
- Quorum Server Messages.

### Serviceguard Command Hangs

Many Serviceguard commands, including `cmviewcl`, depend on name resolution services to look up the addresses of cluster nodes. When name services are not available (for example, if a name server is down), Serviceguard commands may hang, or may return a network-related error message. If this happens, use the `host` command on each cluster node to see whether name resolution is correct. For example:

```
# host ftsys9
ftsys9.cup.hp.com has address 15.13.172.229
```

If the output of this command does not include the correct IP address of the node, then check your name resolution services further.

## Cluster Re-formations

Cluster re-formations may occur from time to time due to current cluster conditions. Some of the causes are as follows:

- local switch on an Ethernet LAN if the switch takes longer than the cluster `NODE_TIMEOUT` value. To prevent this problem, you can increase the cluster `NODE_TIMEOUT` value.
- excessive network traffic on heartbeat LANs. To prevent this, you can use dedicated heartbeat LANs, or LANs with less traffic on them.
- an overloaded system, with too much total I/O and network traffic.
- an improperly configured network, for example, one with a very large routing table.

In these cases, applications continue running, though they might experience a small performance impact during cluster re-formation.

## System Administration Errors

There are a number of errors you can make when configuring Serviceguard that will not show up when you start the cluster. Your cluster can be running, and everything appears to be fine, until there is a hardware or software failure and control of your packages are not transferred to another node as you would have expected.

These are errors caused specifically by errors in the cluster configuration file and package configuration scripts. Examples of these errors include:

- Volume groups not defined on adoptive node.
- Mount point does not exist on adoptive node.
- Network errors on adoptive node (configuration errors).
- User information not correct on adoptive node.

You can use the following commands to check the status of your disks:

- `df` - to see if your package's volume group is mounted.
- `vgsdisplay -v` - to see if all volumes are present.
- `strings /etc/lvmconf/*.conf` - to ensure that the configuration is correct.



- `fdisk -v /dev/sdx` - to display information about a disk.

### Package Control Script Hangs or Failures

When a `RUN_SCRIPT_TIMEOUT` or `HALT_SCRIPT_TIMEOUT` value is set, and the control script hangs, causing the timeout to be exceeded, Serviceguard kills the script and marks the package “Halted.” Similarly, when a package control script fails, Serviceguard kills the script and marks the package “Halted.” In both cases, the following also take place:

- Control of the package will not be transferred.
- The run or halt instructions may not run to completion.
- Global switching will be disabled.
- The current node will be disabled from running the package.

Following such a failure, since the control script is terminated, some of the package’s resources may be left activated. Specifically:

- Volume groups may be left active.
- File systems may still be mounted.
- IP addresses may still be installed.
- Services may still be running.

In this kind of situation, Serviceguard will not restart the package without manual intervention. You must clean up manually before restarting the package. Use the following steps as guidelines:

1. Perform application specific cleanup. Any application specific actions the control script might have taken should be undone to ensure successfully starting the package on an alternate node. This might include such things as shutting down application processes, removing lock files, and removing temporary files.
2. Ensure that package IP addresses are removed from the system. This step is accomplished via the `cmmmodnet (1m)` command. First determine which package IP addresses are installed by inspecting the output resulting from running the `ifconfig` command. If any of the IP addresses specified in the package control script appear in the `ifconfig` output under the “inet addr:” in the “ethX:Y” block, use `cmmmodnet` to remove them:

```
# cmmmodnet -r -i <ip-address> <subnet>
```

where *<ip-address>* is the address indicated above and *<subnet>* is the result of masking the *<ip-address>* with the mask found in the same line as the inet address in the ifconfig output.

3. Ensure that package volume groups are deactivated. First unmount any package logical volumes which are being used for file systems. This is determined by inspecting the output resulting from running the command `df -l`. If any package logical volumes, as specified by the `LV[]` array variables in the package control script, appear under the “Filesystem” column, use `umount` to unmount them:

```
# fuser -ku <logical-volume>
# umount <logical-volume>
```

Next, deactivate the package volume groups. These are specified by the `VG[]` array entries in the package control script.

```
# vgchange -a n <volume-group>
```

4. Finally, re-enable the package for switching.

```
# cmmmodpkg -e <package-name>
```

If after cleaning up the node on which the timeout occurred it is desirable to have that node as an alternate for running the package, remember to re-enable the package to run on the node:

```
# cmmmodpkg -e -n <node-name> <package-name>
```

The default Serviceguard control scripts are designed to take the straightforward steps needed to get an application running or stopped. If the package administrator specifies a time limit within which these steps need to occur and that limit is subsequently exceeded for any reason, Serviceguard takes the conservative approach that the control script logic must either be hung or defective in some way. At that point the control script cannot be trusted to perform cleanup actions correctly, thus the script is terminated and the package administrator is given the opportunity to assess what cleanup steps must be taken.

If you want the package to switch automatically in the event of a control script timeout, set the `NODE_FAIL_FAST_ENABLED` parameter to YES. In this case, Serviceguard will cause a TOC on the node where the control script timed out. This effectively cleans up any side effects of the package’s run or halt attempt. In this case the package will be automatically restarted on any available alternate node for which it is configured.

## Package Movement Errors

These errors are similar to the system administration errors except they are caused specifically by errors in the package control script. The best way to prevent these errors is to test your package control script before putting your high availability application on line.

Adding a “`set -x`” statement in the second line of your control script will give you details on where your script may be failing.

## Node and Network Failures

These failures cause Serviceguard to transfer control of a package to another node. This is the normal action of Serviceguard, but you have to be able to recognize when a transfer has taken place and decide to leave the cluster in its current condition or to restore it to its original condition.

Possible node failures can be caused by the following conditions:

- TOC
- Kernel Oops
- Hangs
- Power failures

You can use the following commands to check the status of your network and subnets:

- `ifconfig` - to display LAN status and check to see if the package IP is stacked on the LAN card.
- `arp -a` - to check the arp tables.

Since your cluster is unique, there are no cookbook solutions to all possible problems. But if you apply these checks and commands and work your way through the log files, you will be successful in identifying and solving problems.

## Quorum Server Messages

The coordinator node in Serviceguard sometimes sends a request to the quorum server to set the lock state. (This is different from a request to obtain the lock in tie-breaking.) If the quorum server's connection to one of the cluster nodes has not completed, the request to set may fail with a two-line message like the following in the quorum server's log file:

```
Oct 008 16:10:05:0: There is no connection to the applicant
2 for lock /sg/lockTest1
Oct 08 16:10:05:0:Request for lock /sg/lockTest1 from
applicant 1 failed: not connected to all applicants.
```

This condition can be ignored. When the failure happens, the QS client within `cmcltd` on the coordinator node does not log anything, but tries the request again within a few seconds. This request succeeds because the connections to all nodes have completed. The following message is logged:

```
Oct 008 16:10:06:0: Request for lock /sg/lockTest1
succeeded. New lock owners: 1,2.
```

## Lock LUN Messages

If the lock LUN device fails, the following message will be entered in the syslog file:

```
Oct 008 16:10:05:0: WARNING: Cluster lock lun /dev/sdc1 has
failed.
```

# A Serviceguard Commands

The following is an alphabetical list of commands used for ServiceGuard cluster configuration and maintenance. Man pages for these commands are available on your system *after installation*.

**Table A-1** Serviceguard Commands

Command	Description
cmapplyconf	<p>Verify and apply ServiceGuard cluster configuration and package configuration files.</p> <p>cmapplyconf verifies the cluster configuration and package configuration specified in the <code>cluster_ascii_file</code> and the associated <code>pkg_ascii_file(s)</code>, creates or updates the binary configuration file, called <code>cmclconfig</code>, and distributes it to all nodes. This binary configuration file contains the cluster configuration information as well as package configuration information for all packages specified. This file, which is used by the cluster daemons to manage the entire cluster and package environment, is kept in the <code>\$SGCONF</code> directory.</p> <p>If changes to either the cluster configuration or to any of the package configuration files are needed, first update the appropriate ASCII file(s) (cluster or package), then validate the changes using the <code>cmcheckconf</code> command and then use <code>cmapplyconf</code> again to verify and redistribute the binary file to all nodes. The cluster and package configuration can be modified only when the cluster is down. The cluster ASCII file only needs to be specified if configuring the cluster for the first time, or if adding or deleting nodes to the cluster. The package ASCII file only needs to be specified if the package is being added, or if the package configuration is being modified.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

<b>Command</b>	<b>Description</b>
cmapplyconf (continued)	<p>It is recommended that the user run the <code>cmgetconf</code> command to get either the cluster ASCII configuration file or package ASCII configuration file whenever changes to the existing configuration are required.</p> <p>Note that <code>cmapplyconf</code> will verify and distribute cluster configuration or package files. It will not cause the cluster daemon to start or removed from the cluster configuration. The same kind of processing will apply to the package configuration to determine whether to add or delete package nodes, package subnet, etc. All package configuration changes require the package to be halted.</p>
cmcheckconf	<p>Check high availability cluster configuration and/or package configuration files.</p> <p><code>cmcheckconf</code> verifies the cluster configuration as specified by the <code>cluster_ascii_file</code> and/or the package configuration files specified by each <code>pkg_ascii_file</code> in the command. If the cluster has already been configured previously, the <code>cmcheckconf</code> command will compare the configuration in the <code>cluster_ascii_file</code> against the previously configuration information stored in the binary configuration file and validates the changes. The same rules apply to the <code>pkg_ascii_file</code>. It is necessary to halt the cluster to run the <code>cmcheckconf</code> command.</p>
cmconfigres	<p>Configure disk monitoring.</p> <p>The command is used to manage the <code>cmresmond</code> configuration file. The default path of this file is <code>\$SGCONF/cmresmond_config.xml</code>.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

Command	Description
cmdeleteconf	<p>Delete either the cluster or the package configuration.</p> <p>cmdeleteconf deletes either the entire cluster configuration, including all its packages, or only the specified package configuration. If neither <i>cluster_name</i> nor <i>package_name</i> is specified, cmdeleteconf will delete the local cluster's configuration and all its packages. If only the <i>package_name</i> is specified, the configuration of <i>package_name</i> in the local cluster is deleted. If both <i>cluster_name</i> and <i>package_name</i> are specified, the package must be configured in the <i>cluster_name</i>, and only the package <i>package_name</i> will be deleted. The local cluster is the cluster that the node running the cmdeleteconf command belongs to.</p>
cmgetconf	<p>Get cluster or package configuration information.</p> <p>cmgetconf obtains either the cluster configuration, not including the package configuration, or the specified package's configuration information, and writes to either the <i>output_filename</i> file, or to stdout. This command can be run whether the cluster is up or down. If neither <i>cluster_name</i> nor <i>package_name</i> is specified, cmgetconf will obtain the local cluster's configuration. If both <i>cluster_name</i> and <i>package_name</i> are specified, the package must be configured in the <i>cluster_name</i>, and only the package configuration for <i>package_name</i> will be written to <i>output_filename</i> or to stdout.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

Command	Description
cmhaltcl	<p>Halt a high availability cluster.</p> <p>cmhaltcl causes all nodes in a configured cluster to stop their cluster daemons, optionally halting all packages or applications in the process.</p> <p>This command will halt all the daemons on all currently running systems. If the user only wants to shutdown a subset of daemons, the cmhaltnode command should be used instead.</p>
cmhaltnode	<p>Halt a node in a high availability cluster.</p> <p>cmhaltnode causes a node to halt its cluster daemon and remove itself from the existing cluster.</p> <p>When cmhaltnode is run on a node, the cluster daemon is halted and, optionally, all packages that were running on that node are moved to other nodes if possible.</p> <p>If <i>node_name</i> is not specified, the cluster daemon running on the local node will be halted and removed from the existing cluster.</p>
cmhaltpkg	<p>Halt a high availability package.</p> <p>cmhaltpkg performs a manual halt of high availability package(s) running on ServiceGuard clusters. This command may be run on any node within the cluster and may operate on any package within the cluster.</p> <p>Note: Serviceguard disables AUTO_RUN automatically whenever the command cmhaltpkg is executed. The user needs to re-enable AUTO_RUN using the cmrunpkg -e &lt;pkg&gt; command.</p>



**Table A-1**      **Serviceguard Commands (Continued)**

Command	Description
cmhaltserv	<p>Halt a service from the high availability package halt script. This is not a command line executable command, it runs only from within the package control script.</p> <p>cmhaltserv is used in the high availability package halt script to halt a service. If any part of package is marked down, the package halt script is executed as part of the recovery process.</p> <p>This command sends a SIGTERM signal to the PID and the corresponding process group of the monitored service. If this signal is caught by the running application, it is up to the application to ensure that the processes will be terminated.</p>
cmmakepkg	<p>Create a high availability package template file.</p> <p>cmmakepkg creates a template ASCII package configuration file or package control script as specified by the selected option. The <i>output_file_name</i> should be customized for a specific cluster environment. After customization, these files should be verified by the cmcheckconf command. If <i>output_file_name</i> is not provided, output will be directed to stdout.</p>

**Table A-1** Serviceguard Commands (Continued)

Command	Description
cmmodnet	<p>Add or remove an address from a high availability cluster.</p> <p>cmmodnet is used in the high availability package control scripts to add or remove an <i>IP_address</i> from the current network interface running the given <i>subnet_name</i>.</p> <p>Extreme caution should be exercised when executing this command outside the context of the package control script. In this capacity it should only be used to remove the re-locatable IP addresses of packages which have failed and are in the “halted” state. Use while the package is running could lead to loss of client connectivity.</p>
cmmodpkg	<p>Enable or disable switching attributes for a high availability package.</p> <p>cmmodpkg enables or disables the ability of a package to switch to another node upon failure of the package, and it enables or disables a particular node from running specific packages. Switching for a package can be enabled or disabled globally. For example, if a globally disabled package fails, it will not switch to any other node, and if a globally enabled package fails, it will attempt to switch to the first available node on which it is configured to run.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

<b>Command</b>	<b>Description</b>
<code>cmquerycl</code>	<p>Query cluster or node configuration information.</p> <p><code>cmquerycl</code> searches all specified nodes for cluster configuration, cluster lock, and Logical Volume Manager (LVM) information. Cluster configuration information includes network information such as LAN interface, IP addresses, bridged networks and possible heartbeat networks. Cluster lock information is either the quorum server hostname and IP address or the lock LUN device file data. LVM information includes volume group (VG) interconnection and file system mount point information. This command should be run as the first step in preparing for cluster configuration. It may also be used as a troubleshooting tool to identify the current configuration of a cluster.</p>
<code>cmreadlog</code>	<p>Format an Object Manager log file for easy display.</p> <p>This command reads the log files created by Object Manager in the Managed Object File (MOF) format and displays them in a report with one entry per line. Use the command when troubleshooting or reviewing Object Manager activity.</p>
<code>cmresserviced</code>	<p>Request monitoring of a device from the monitor server, <code>cmresmond</code>. This command is used in the <code>SERVICE_CMD</code> parameter of the package control script to define package dependencies on monitored disks.</p>
<code>cmruncl</code>	<p>Run a high availability cluster.</p> <p><code>cmruncl</code> causes all nodes in a configured cluster or all nodes specified to start their cluster daemons and form a new cluster. This command should only be run when the cluster is not active on any of the configured nodes. If a cluster is already running on a subset of the nodes, the <code>cmrunnode</code> command should be used to start the remaining nodes and force them to join the existing cluster.</p>

**Table A-1**      Serviceguard Commands (Continued)

Command	Description
cmrunnode	<p>Run a node in a high availability cluster.</p> <p>cmrunnode causes a node to start its cluster daemon to join the existing cluster</p> <p>Starting a node will not cause any active packages to be moved to the new node. However, if a package is DOWN, has its switching enabled, and is able to run on the new node, that package will automatically run there.</p>
cmrunpkg	<p>Run a high availability package.</p> <p>cmrunpkg runs a high availability package(s) that was previously halted. This command may be run on any node within the cluster and may operate on any package within the cluster. If a node is not specified, the node on which the command is run will be used. This will result in an error if the current node is not able to run the package or is not in the list of possible owners of the package. When a package is started on a new node, the package's run script is executed.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

Command	Description
<code>cmrunserv</code>	<p>Run a service from the high availability package run script. This is not a command line executable command, it runs only from within the package control script.</p> <p><code>cmrunserv</code> is used in the high availability package run script to run a service. If the service process dies, <code>cmrunserv</code> updates the status of the service to down. The cluster software will recognize the change in status and execute the normal package recovery sequence. This includes executing the package halt script, determining if the package can be run on a different node, and if so, executing the package run script on the new node.</p> <p>Should the <i>service_command</i> be halted by the <code>cmhaltserv</code> command, a SIGTERM signal will be sent to the process. This executable or shell script should be able to handle a SIGTERM signal and execute a graceful shutdown performing any cleanup necessary. If the process ignores the SIGTERM, a SIGKILL will be sent to the process. If a SIGKILL is sent, the process will die immediately and will be unable to perform any cleanup.</p>

**Table A-1** Serviceguard Commands (Continued)

Command	Description
cmscancl	<p>Gather system configuration information from nodes with ServiceGuard installed.</p> <p>cmscancl is a configuration report and diagnostic tool which gathers system software and hardware configuration information from a list of nodes, or from all the nodes in a cluster. The information that this command displays includes LAN device configuration, network status and interfaces, file systems, LVM configuration, link-level connectivity, and the data from the binary cluster configuration file. This command can be used as a troubleshooting tool or as a data collection tool.</p> <p>If output _file is not specified, the information will be directed to stdout. Output file contains:</p> <ul style="list-style-type: none"> <li>• network status and interfaces (output from ifconfig)</li> <li>• file systems (output from mount)</li> <li>• LVM configuration (contents of /etc/lvmtab file)</li> <li>• ip-level connectivity (output from ping)</li> <li>• binary configuration file data (output from cmviewconf)</li> </ul>
cmviewcl	<p>View information about the current high availability cluster.</p> <p>cmviewcl displays the current status information of a cluster. Output can be displayed for the whole cluster or it may be limited to particular nodes or packages.</p>

**Table A-1**      **Serviceguard Commands (Continued)**

Command	Description
cmviewconf	<p>View Serviceguard or ServiceGuard cluster configuration information.</p> <p>cmviewconf collects and displays the cluster configuration information, in ASCII format, from the binary configuration file for an existing cluster. Optionally, the output can be written to a file. This command can be used as a troubleshooting tool to identify the configuration of a cluster.</p>





# B

## Designing Highly Available Cluster Applications

This appendix describes how to create or port applications for high availability, with emphasis on the following topics:

- Automating Application Operation
- Controlling the Speed of Application Failover
- Designing Applications to Run on Multiple Systems
- Restoring Client Connections
- Handling Application Failures
- Minimizing Planned Downtime

Designing for high availability means reducing the amount of unplanned and planned downtime that users will experience. Unplanned downtime includes unscheduled events such as power outages, system failures, network failures, disk crashes, or application failures. Planned downtime includes scheduled events such as scheduled backups, system upgrades to new OS revisions, or hardware replacements.

Two key strategies should be kept in mind:

1. Design the application to handle a system reboot or panic. If you are modifying an existing application for a highly available environment, determine what happens currently with the application after a system panic. In a highly available environment there should be defined (and scripted) procedures for restarting the application. Procedures for starting and stopping the application should be automatic, with no user intervention required.
2. The application should not use any system-specific information such as the following if such use would prevent it from failing over to another system and running properly:
  - The application should not refer to `uname()` or `gethostname()`.
  - The application should not refer to the SPU ID.
  - The application should not refer to the MAC (link-level) address.

## **Automating Application Operation**

Can the application be started and stopped automatically or does it require operator intervention?

This section describes how to automate application operations to avoid the need for user intervention. One of the first rules of high availability is to avoid manual intervention. If it takes a user at a terminal, console or GUI interface to enter commands to bring up a subsystem, the user becomes a key part of the system. It may take hours before a user can get to a system console to do the work necessary. The hardware in question may be located in a far-off area where no trained users are available, the systems may be located in a secure datacenter, or in off hours someone may have to connect via modem.

There are two principles to keep in mind for automating application relocation:

- Insulate users from outages.
- Applications must have defined startup and shutdown procedures.

You need to be aware of what happens currently when the system your application is running on is rebooted, and whether changes need to be made in the application's response for high availability.

### **Insulate Users from Outages**

Wherever possible, insulate your end users from outages. Issues include the following:

- Do not require user intervention to reconnect when a connection is lost due to a failed server.
- Where possible, warn users of slight delays due to a failover in progress.
- Minimize the reentry of data.
- Engineer the system for reserve capacity to minimize the performance degradation experienced by users.

## Define Application Startup and Shutdown

Applications must be restartable without manual intervention. If the application requires a switch to be flipped on a piece of hardware, then automated restart is impossible. Procedures for application startup, shutdown and monitoring must be created so that the HA software can perform these functions automatically.

To ensure automated response, there should be defined procedures for starting up the application and stopping the application. In Serviceguard these procedures are placed in the package control script. These procedures must check for errors and return status to the HA control software. The startup and shutdown should be command-line driven and not interactive unless all of the answers can be predetermined and scripted.

In an HA failover environment, HA software restarts the application on a surviving system in the cluster that has the necessary resources, like access to the necessary disk drives. The application must be restartable in two aspects:

- It must be able to restart and recover on the backup system (or on the same system if the application restart option is chosen).
- It must be able to restart if it fails during the startup and the cause of the failure is resolved.

Application administrators need to learn to startup and shutdown applications using the appropriate HA commands. Inadvertently shutting down the application directly will initiate an unwanted failover. Application administrators also need to be careful that they don't accidentally shut down a production instance of an application rather than a test instance in a development environment.

A mechanism to monitor whether the application is active is necessary so that the HA software knows when the application has failed. This may be as simple as a script that issues the command `ps -ef | grep xxx` for all the processes belonging to the application.

To reduce the impact on users, the application should not simply abort in case of error, since aborting would cause an unneeded failover to a backup system. Applications should determine the exact error and take specific action to recover from the error rather than, for example, aborting upon receipt of any error.

## Controlling the Speed of Application Failover

What steps can be taken to ensure the fastest failover?

If a failure does occur causing the application to be moved (failed over) to another node, there are many things the application can do to reduce the amount of time it takes to get the application back up and running. The topics covered are as follows:

- Replicate Non-Data File Systems
- Use Raw Volumes
- Evaluate the Use of a journaled file system
- Minimize Data Loss
- Use Restartable Transactions
- Use Checkpoints
- Design for Multiple Servers
- Design for Replicated Data Sites

### Replicate Non-Data File Systems

Non-data file systems should be replicated rather than shared. There can only be one copy of the application data itself. It will be located on a set of disks that is accessed by the system that is running the application. After failover, if these data disks are filesystems, they must go through filesystems recovery (`fsck`) before the data can be accessed. To help reduce this recovery time, the smaller these filesystems are, the faster the recovery will be. Therefore, it is best to keep anything that can be replicated off the data filesystem. For example, there should be a copy of the application executables on each system rather than having one copy of the executables on a shared filesystem. Additionally, replicating the application executables makes them subject to a rolling upgrade if this is desired.

## **Evaluate the Use of a Journalled Filesystem (JFS)**

If a file system must be used, a JFS offers significantly faster file system recovery than an HFS. However, performance of the JFS may vary with the application. An example of an appropriate JFS is the Reiser FS or ext3.

## **Minimize Data Loss**

Minimize the amount of data that might be lost at the time of an unplanned outage. It is impossible to prevent some data from being lost when a failure occurs. However, it is advisable to take certain actions to minimize the amount of data that will be lost, as explained in the following discussion.

### **Minimize the Use and Amount of Memory-Based Data**

Any in-memory data (the in-memory context) will be lost when a failure occurs. The application should be designed to minimize the amount of in-memory data that exists unless this data can be easily recalculated. When the application restarts on the standby node, it must recalculate or reread from disk any information it needs to have in memory.

One way to measure the speed of failover is to calculate how long it takes the application to start up on a normal system after a reboot. Does the application start up immediately? Or are there a number of steps the application must go through before an end-user can connect to it? Ideally, the application can start up quickly without having to reinitialize in-memory data structures or tables.

Performance concerns might dictate that data be kept in memory rather than written to the disk. However, the risk associated with the loss of this data should be weighed against the performance impact of posting the data to the disk.

Data that is read from a shared disk into memory, and then used as read-only data can be kept in memory without concern.

### **Keep Logs Small**

Some databases permit logs to be buffered in memory to increase online performance. Of course, when a failure occurs, any in-flight transaction will be lost. However, minimizing the size of this in-memory log will reduce the amount of completed transaction data that would be lost in case of failure.

Keeping the size of the on-disk log small allows the log to be archived or replicated more frequently, reducing the risk of data loss if a disaster were to occur. There is, of course, a trade-off between online performance and the size of the log.

### **Eliminate Need for Local Data**

When possible, eliminate the need for local data. In a three-tier, client/server environment, the middle tier can often be dataless (i.e., there is no local data that is client specific or needs to be modified). This “application server” tier can then provide additional levels of availability, load-balancing, and failover. However, this scenario requires that all data be stored either on the client (tier 1) or on the database server (tier 3).

### **Use Restartable Transactions**

Transactions need to be restartable so that the client does not need to re-enter or back out of the transaction when a server fails, and the application is restarted on another system. In other words, if a failure occurs in the middle of a transaction, there should be no need to start over again from the beginning. This capability makes the application more robust and reduces the visibility of a failover to the user.

A common example is a print job. Printer applications typically schedule jobs. When that job completes, the scheduler goes on to the next job. If, however, the system dies in the middle of a long job (say it is printing paychecks for 3 hours), what happens when the system comes back up again? Does the job restart from the beginning, reprinting all the paychecks, does the job start from where it left off, or does the scheduler assume that the job was done and not print the last hours worth of paychecks? The correct behavior in a highly available environment is to restart where it left off, ensuring that everyone gets one and only one paycheck.

Another example is an application where a clerk is entering data about a new employee. Suppose this application requires that employee numbers be unique, and that after the name and number of the new employee is entered, a failure occurs. Since the employee number had been entered before the failure, does the application refuse to allow it to be re-entered? Does it require that the partially entered information be deleted first? More appropriately, in a highly available environment the application will allow the clerk to easily restart the entry or to continue at the next data item.

## **Use Checkpoints**

Design applications to checkpoint complex transactions. A single transaction from the user's perspective may result in several actual database transactions. Although this issue is related to restartable transactions, here it is advisable to record progress locally on the client so that a transaction that was interrupted by a system failure can be completed after the failover occurs.

For example, suppose the application being used is calculating PI. On the original system, the application has gotten to the 1,000th decimal point, but the application has not yet written anything to disk. At that moment in time, the node crashes. The application is restarted on the second node, but the application is started up from scratch. The application must recalculate those 1,000 decimal points. However, if the application had written to disk the decimal points on a regular basis, the application could have restarted from where it left off.

## **Balance Checkpoint Frequency with Performance**

It is important to balance checkpoint frequency with performance. The trade-off with checkpointing to disk is the impact of this checkpointing on performance. Obviously if you checkpoint too often the application slows; if you don't checkpoint often enough, it will take longer to get the application back to its current state after a failover. Ideally, the end-user should be able to decide how often to checkpoint. Applications should provide customizable parameters so the end-user can tune the checkpoint frequency.

## Design for Multiple Servers

If you use multiple active servers, multiple service points can provide relatively transparent service to a client. However, this capability requires that the client be smart enough to have knowledge about the multiple servers and the priority for addressing them. It also requires access to the data of the failed server or replicated data.

For example, rather than having a single application which fails over to a second system, consider having both systems running the application. After a failure of the first system, the second system simply takes over the load of the first system. This eliminates the start up time of the application. There are many ways to design this sort of architecture, and there are also many issues with this sort of design. This discussion will not go into details other than to give a few examples.

The simplest method is to have two applications running in a master/slave relationship where the slave is simply a hot standby application for the master. When the master fails, the slave on the second system would still need to figure out what state the data was in (i.e., data recovery would still take place). However, the time to fork the application and do the initial startup is saved.

Another possibility is having two applications that are both active. An example might be two application servers which feed a database. Half of the clients connect to one application server and half of the clients connect to the second application server. If one server fails, then all the clients connect to the remaining application server.

## Design for Replicated Data Sites

Replicated data sites are a benefit for both fast failover and disaster recovery. With replicated data, data disks are *not* shared between systems. There is no data recovery that has to take place. This makes the recovery time faster. However, there may be performance trade-offs associated with replicating data. There are a number of ways to perform data replication, which should be fully investigated by the application designer.

Many of the standard database products provide for data replication transparent to the client application. By designing your application to use a standard database, the end-user can determine if data replication is desired.



## Designing Applications to Run on Multiple Systems

If an application can be failed to a backup node, how will it work on that different system?

The previous sections discussed methods to ensure that an application can be automatically restarted. This section will discuss some ways to ensure the application can run on multiple systems. Topics are as follows:

- Avoid Node Specific Information
- Assign Unique Names to Applications
- Use `Uname(2)` With Care
- Bind to a Fixed Port
- Bind to a Relocatable IP Addresses
- Give Each Application its Own Volume Group
- Use Multiple Destinations for SNA Applications
- Avoid File Locking

### Avoid Node Specific Information

Typically, when a new system is installed, an IP address must be assigned to each active network interface. This IP address is always associated with the node and is called a **stationary** IP address.

The use of packages containing highly available applications adds the requirement for an additional set of IP addresses, which are assigned to the applications themselves. These are known as **relocatable** application IP addresses. Serviceguard's network sensor monitors the node's access to the subnet on which these relocatable application IP addresses reside. When packages are configured in Serviceguard, the associated subnetwork address is specified as a package dependency, and a list of nodes on which the package can run is also provided. When failing a package over to a remote node, the subnetwork must already be active on the target node.

Each application or package should be given a unique name as well as a relocatable IP address. Following this rule separates the application from the system on which it runs, thus removing the need for user knowledge of which system the application runs on. It also makes it easier to move the application among different systems in a cluster for load balancing or other reasons. If two applications share a single IP address, they must move together. Instead, using independent names and addresses allows them to move separately.

For external access to the cluster, clients must know how to refer to the application. One option is to tell the client which relocatable IP address is associated with the application. Another option is to think of the application name as a host, and configure a name-to-address mapping in the Domain Name System (DNS). In either case, the client will ultimately be communicating via the application's relocatable IP address. If the application moves to another node, the IP address will move with it, allowing the client to use the application without knowing its current location. Remember that each network interface must have a stationary IP address associated with it. This IP address does *not* move to a remote system in the event of a network failure.

### **Obtain Enough IP Addresses**

Each application receives a *relocatable* IP address that is separate from the stationary IP address assigned to the system itself. Therefore, a single system might have many IP addresses, one for itself and one for each of the applications that it normally runs. Therefore, IP addresses in a given subnet range will be consumed faster than without high availability. It might be necessary to acquire additional IP addresses.

Multiple IP addresses on the same network interface are supported only if they are on the same subnetwork.

### **Allow Multiple Instances on Same System**

Applications should be written so that multiple instances, each with its own application name and IP address, can run on a single system. It might be necessary to invoke the application with a parameter showing which instance is running. This allows distributing the users among several systems under normal circumstances, but it also allows all of the users to be serviced in the case of a failure on a single system.

## Avoid Using SPU IDs or MAC Addresses

Design the application so that it does not rely on the SPU ID or MAC (link-level) addresses. The SPU ID is a unique hardware ID contained in non-volatile memory, which cannot be changed. A MAC address (also known as a NIC id) is a link-specific address associated with the LAN hardware. The use of these addresses is a common problem for license servers, since for security reasons they want to use hardware-specific identification to ensure the license isn't copied to multiple nodes. One workaround is to have multiple licenses; one for each node the application will run on. Another way is to have a cluster-wide mechanism that lists a set of SPU IDs or node names. If your application is running on a system in the specified set, then the license is approved.

Previous generation HA software would move the MAC address of the network card along with the IP address when services were moved to a backup system. This is no longer allowed in Serviceguard.

There were a couple of reasons for using a MAC address, which have been addressed below:

- Old network devices between the source and the destination such as routers had to be manually programmed with MAC and IP address pairs. The solution to this problem is to move the MAC address along with the IP address in case of failover.
- Up to 20 minute delays could occur while network device caches were updated due to timeouts associated with systems going down. This is dealt with in current HA software by broadcasting a new ARP translation of the old IP address with the new MAC address.

## Assign Unique Names to Applications

A unique name should be assigned to each application. This name should then be configured in DNS so that the name can be used as input to `gethostbyname(3)`, as described in the following discussion.

### Use DNS

DNS provides an API which can be used to map hostnames to IP addresses and vice versa. This is useful for BSD socket applications such as telnet which are first told the target system name. The application must then map the name to an IP address in order to establish a connection. However, some calls should be used with caution.

Applications should *not* reference official hostnames or IP addresses. The official hostname and corresponding IP address for the hostname refer to the primary LAN card and the *stationary IP address* for that card. Therefore, any application that refers to, or requires the hostname or primary IP address may not work in an HA environment where the network identity of the system that supports a given application moves from one system to another, but the hostname does not move.

One way to look for problems in this area is to look for calls to `gethostname(2)` in the application. HA services should use `gethostname()` with caution, since the response may change over time if the application migrates. Applications that use `gethostname()` to determine the name for a call to `gethostbyname(3)` should also be avoided for the same reason. Also, the `gethostbyaddr()` call may return different answers over time if called with a stationary IP address.

Instead, the application should always refer to the application name and relocatable IP address rather than the hostname and stationary IP address. It is appropriate for the application to call `gethostbyname(3)`, specifying the application name rather than the hostname. `gethostbyname(3)` will pass in the IP address of the application. This IP address will move with the application to the new node.

However, `gethostbyname(3)` should be used to locate the IP address of an application only if the application name is configured in DNS. It is probably best to associate a different application name with each independent HA service. This allows each application and its IP address to be moved to another node without affecting other applications. Only the stationary IP addresses should be associated with the hostname in DNS.

## **Use `uname(2)` With Care**

Related to the `hostname` issue discussed in the previous section is the application's use of `uname(2)`, which returns the official system name. The system name is unique to a given system whatever the number of LAN cards in the system. By convention, the `uname` and `hostname` are the same, but they do not have to be. Some applications, after connection to a system, might call `uname(2)` to validate for security purposes that they are really on the correct system. This is not appropriate in an HA environment, since the service is moved from one system to another, and neither the `uname` nor the `hostname` are moved. Applications should develop alternate means of verifying where they are running. For example, an application might check a list of hostnames that have been provided in a configuration file.

## Bind to a Fixed Port

When binding a socket, a port address can be specified or one can be assigned dynamically. One issue with binding to random ports is that a different port may be assigned if the application is later restarted on another cluster node. This may be confusing to clients accessing the application.

The recommended method is using fixed ports that are the same on all nodes where the application will run, instead of assigning port numbers dynamically. The application will then always return the same port number regardless of which node is currently running the application. Application port assignments should be put in `/etc/services` to keep track of them and to help ensure that someone will not choose the same port number.

## Bind to Relocatable IP Addresses

When sockets are bound, an IP address is specified in addition to the port number. This indicates the IP address to use for communication and is meant to allow applications to limit which interfaces can communicate with clients. An application can bind to `INADDR_ANY` as an indication that messages can arrive on any interface.

Network applications can bind to a stationary IP address, a relocatable IP address, or `INADDR_ANY`. If the stationary IP address is specified, then the application may fail when restarted on another node, because the stationary IP address is not moved to the new system. If an application binds to the relocatable IP address, then the application will behave correctly when moved to another system.

Many server-style applications will bind to `INADDR_ANY`, meaning that they will receive requests on any interface. This allows clients to send to the stationary or relocatable IP addresses. However, in this case the networking code cannot determine which source IP address is most appropriate for responses, so it will always pick the stationary IP address.

For TCP stream sockets, the TCP level of the protocol stack resolves this problem for the client since it is a connection-based protocol. On the client, TCP ignores the stationary IP address and continues to use the previously bound relocatable IP address originally used by the client.

With UDP datagram sockets, however, there is a problem. The client may connect to multiple servers utilizing the relocatable IP address and sort out the replies based on the source IP address in the server's response message. However, the source IP address given in this response will be the stationary IP address rather than the relocatable application IP address. Therefore, when creating a UDP socket for listening, the application must always call `bind(2)` with the appropriate relocatable application IP address rather than `INADDR_ANY`.

If the application cannot be modified as recommended above, a workaround to this problem is to not use the stationary IP address at all, and only use a single relocatable application IP address on a given LAN card. Limitations with this workaround are as follows:

- Local LAN failover will not work.
- There has to be an idle LAN card on each backup node that is used to relocate the relocatable application IP address in case of a failure.

### **Call `bind()` before `connect()`**

When an application initiates its own connection, it should first call `bind(2)`, specifying the application IP address before calling `connect(2)`. Otherwise the connect request will be sent using the stationary IP address of the system's outbound LAN interface rather than the desired relocatable application IP address. The client will receive this IP address from the `accept(2)` call, possibly confusing the client software and preventing it from working correctly.

## Give Each Application its Own Volume Group

Use separate volume groups for each application that uses data. If the application doesn't use disk, it is not necessary to assign it a separate volume group. A volume group (group of disks) is the unit of storage that can move between nodes. The greatest flexibility for load balancing exists when each application is confined to its own volume group, i.e., two applications do not share the same set of disk drives. If two applications do use the same volume group to store their data, then the applications must move together. If the applications' data stores are in separate volume groups, they can switch to different nodes in the event of a failover.

The application data should be set up on different disk drives and if applicable, different mount points. The application should be designed to allow for different disks and separate mount points. If possible, the application should not assume a specific mount point.

## Use Multiple Destinations for SNA Applications

SNA is point-to-point link-oriented; that is, the *services* cannot simply be moved to another system, since that system has a different point-to-point link which originates in the mainframe. Therefore, backup links in a node and/or backup links in other nodes should be configured so that SNA does not become a single point of failure. Note that only one configuration for an SNA link can be active at a time. Therefore, backup links that are used for other purposes should be reconfigured for the primary mission-critical purpose upon failover.



## **Avoid File Locking**

In an NFS environment, applications should avoid using file-locking mechanisms, where the file to be locked is on an NFS Server. File locking should be avoided in an application both on local and remote systems. If local file locking is employed and the system fails, the system acting as the backup system will not have any knowledge of the locks maintained by the failed system. This may or may not cause problems when the application restarts.

Remote file locking is the worst of the two situations, since the system doing the locking may be the system that fails. Then, the lock might never be released, and other parts of the application will be unable to access that data. In an NFS environment, file locking can cause long delays in case of NFS client system failure and might even delay the failover itself.

## Restoring Client Connections

How does a client reconnect to the server after a failure?

It is important to write client applications to specifically differentiate between the loss of a connection to the server and other application-oriented errors that might be returned. The application should take special action in case of connection loss.

One question to consider is how a client knows after a failure when to reconnect to the newly started server. The typical scenario is that the client must simply restart their session, or relog in. However, this method is not very automated. For example, a well-tuned hardware and application system may fail over in 5 minutes. But if users, after experiencing no response during the failure, give up after 2 minutes and go for coffee and don't come back for 28 minutes, the perceived downtime is actually 30 minutes, not 5. Factors to consider are the number of reconnection attempts to make, the frequency of reconnection attempts, and whether or not to notify the user of connection loss.

There are a number of strategies to use for client reconnection:

- Design clients which continue to try to reconnect to their failed server.

Put the work into the client application rather than relying on the user to reconnect. If the server is back up and running in 5 minutes, and the client is continually retrying, then after 5 minutes, the client application will reestablish the link with the server and either restart or continue the transaction. No intervention from the user is required.

- Design clients to reconnect to a *different* server.

If you have a server design which includes multiple active servers, the client could connect to the second server, and the user would only experience a brief delay.

The problem with this design is knowing when the client should switch to the second server. How long does a client retry to the first server before giving up and going to the second server? There are no definitive answers for this. The answer depends on the design of the server application. If the application can be restarted on the same node after a failure (see “Handling Application Failures” following),

the retry to the current server should continue for the amount of time it takes to restart the server locally. This will keep the client from having to switch to the second server in the event of a application failure.

- Use a transaction processing monitor or message queueing software to increase robustness.

Use transaction processing monitors such as Tuxedo or DCE/Encina, which provide an interface between the server and the client.

Transaction processing monitors (TPMs) can be useful in creating a more highly available application. Transactions can be queued such that the client does not detect a server failure. Many TPMs provide for the optional automatic rerouting to alternate servers or for the automatic retry of a transaction. TPMs also provide for ensuring the reliable completion of transactions, although they are not the only mechanism for doing this. After the server is back online, the transaction monitor reconnects to the new server and continues routing it the transactions.

- **Queue Up Requests**

As an alternative to using a TPM, queue up requests when the server is unavailable. Rather than notifying the user when a server is unavailable, the user request is queued up and transmitted later when the server becomes available again. Message queueing software ensures that messages of any kind, not necessarily just transactions, are delivered and acknowledged.

Message queueing is useful only when the user does not need or expect response that the request has been completed (i.e, the application is not interactive).

## Handling Application Failures

What happens if part or all of an application fails?

All of the preceding sections have assumed the failure in question was not a failure of the application, but of another component of the cluster. This section deals specifically with application problems. For instance, software bugs may cause an application to fail or system resource issues (such as low swap/memory space) may cause an application to die. The section deals with how to design your application to recover after these types of failures.

### Create Applications to be Failure Tolerant

An application should be tolerant to failure of a single component. Many applications have multiple processes running on a single node. If one process fails, what happens to the other processes? Do they also fail? Can the failed process be restarted on the same node without affecting the remaining pieces of the application?

Ideally, if one process fails, the other processes can wait a period of time for that component to come back online. This is true whether the component is on the same system or a remote system. The failed component can be restarted automatically on the same system and rejoin the waiting processing and continue on. This type of failure can be detected and restarted within a few seconds, so the end user would never know a failure occurred.

Another alternative is for the failure of one component to still allow bringing down the other components cleanly. If a database SQL server fails, the database should still be able to be brought down cleanly so that no database recovery is necessary.

The worse case is for a failure of one component to cause the entire system to fail. If one component fails and all other components need to be restarted, the downtime will be high.

## **Be Able to Monitor Applications**

All components in a system, including applications, should be able to be monitored for their health. A monitor might be as simple as a display command or as complicated as a SQL query. There must be a way to ensure that the application is behaving correctly. If the application fails and it is not detected automatically, it might take hours for a user to determine the cause of the downtime and recover from it.

## Minimizing Planned Downtime

Planned downtime (as opposed to unplanned downtime) is scheduled; examples include backups, systems upgrades to new operating system revisions, or hardware replacements. For planned downtime, application designers should consider:

- **Reducing the time needed for application upgrades/patches.**

Can an administrator install a new version of the application without scheduling downtime? Can different revisions of an application operate within a system? Can different revisions of a client and server operate within a system?

- **Providing for online application reconfiguration.**

Can the configuration information used by the application be changed without bringing down the application?

- **Documenting maintenance operations.**

Does an operator know how to handle maintenance operations?

When discussing highly available systems, unplanned failures are often the main point of discussion. However, if it takes 2 weeks to upgrade a system to a new revision of software, there are bound to be a large number of complaints.

The following sections discuss ways of handling the different types of planned downtime.

## **Reducing Time Needed for Application Upgrades and Patches**

Once a year or so, a new revision of an application is released. How long does it take for the end-user to upgrade to this new revision? This answer is the amount of planned downtime a user must take to upgrade their application. The following guidelines reduce this time.

### **Provide for Rolling Upgrades**

Provide for a “rolling upgrade” in a client/server environment. For a system with many components, the typical scenario is to bring down the entire system, upgrade every node to the new version of the software, and then restart the application on all the affected nodes. For large systems, this could result in a long downtime. An alternative is to provide for a rolling upgrade. A rolling upgrade rolls out the new software in a phased approach by upgrading only one component at a time. For example, the database server is upgraded on Monday, causing a 15 minute downtime. Then on Tuesday, the application server on two of the nodes is upgraded, which leaves the application servers on the remaining nodes online and causes no downtime. On Wednesday, two more application servers are upgraded, and so on. With this approach, you avoid the problem where everything changes at once, plus you minimize long outages.

The trade-off is that the application software must operate with different revisions of the software. In the above example, the database server might be at revision 5.0 while the some of the application servers are at revision 4.0. The application must be designed to handle this type of situation.

### **Do Not Change the Data Layout Between Releases**

Migration of the data to a new format can be very time intensive. It also almost guarantees that rolling upgrade will not be possible. For example, if a database is running on the first node, ideally, the second node could be upgraded to the new revision of the database. When that upgrade is completed, a brief downtime could be scheduled to move the database server from the first node to the newly upgraded second node. The database server would then be restarted, while the first node is idle and ready to be upgraded itself. However, if the new database revision requires a different database layout, the *old* data will not be readable by the newly updated database. The downtime will be longer as the data is migrated to the new layout.

### **Providing Online Application Reconfiguration**

Most applications have some sort of configuration information that is read when the application is started. If to make a change to the configuration, the application must be halted and a new configuration file read, downtime is incurred.

To avoid this downtime use configuration tools that interact with an application and make dynamic changes online. The ideal solution is to have a configuration tool which interacts with the application. Changes are made online with little or no interruption to the end-user. This tool must be able to do everything online, such as expanding the size of the data, adding new users to the system, adding new users to the application, etc. Every task that an administrator needs to do to the application system can be made available online.

### **Documenting Maintenance Operations**

Standard procedures are important. An application designer should make every effort to make tasks common for both the highly available environment and the normal environment. If an administrator is accustomed to bringing down the entire system after a failure, he or she will continue to do so even if the application has been redesigned to handle a single failure. It is important that application documentation discuss alternatives with regards to high availability for typical maintenance operations.



# C

## Integrating HA Applications with Serviceguard

The following is a summary of the steps you should follow to integrate an application into the Serviceguard environment:

1. Read the rest of this book, including the chapters on cluster and package configuration, and the appendix “Designing Highly Available Cluster Applications.”
2. Define the cluster’s behavior for normal operations:
  - What should the cluster look like during normal operation?
  - What is the standard configuration most people will use? (Is there any data available about user requirements?)
  - Can you separate out functions such as database or application server onto separate machines, or does everything run on one machine?
3. Define the cluster’s behavior for failover operations:
  - Does everything fail over together to the adoptive node?
  - Can separate applications fail over to the same node?
  - Is there already a high availability mechanism within the application other than the features provided by Serviceguard?
4. Identify problem areas
  - What does the application do today to handle a system reboot or panic?
  - Does the application use any system-specific information such as `uname()` or `gethostname()`, `SPU_ID` or MAC address which would prevent it from failing over to another system?

---

## **Checklist for Integrating HA Applications**

This section contains a checklist for integrating HA applications in both single and multiple systems.

### **Defining Baseline Application Behavior on a Single System**

1. Define a baseline behavior for the application on a standalone system:
  - Install the application, database, and other required resources on one of the systems. Be sure to follow Serviceguard rules in doing this:
    - Install all shared data on separate external volume groups.
    - Use a journaled filesystem (JFS) as appropriate.
  - Perform some sort of standard test to ensure the application is running correctly. This test can be used later in testing with Serviceguard. If possible, try to connect to the application through a client.
  - Crash the standalone system, reboot it, and test how the application starts up again. Note the following:
    - Are there any manual procedures? if so, document them.
    - Can everything start up from rc scripts?
  - Try to write a simple script which brings everything up without having to do any keyboard typing. Figure out what the administrator would do at the keyboard, then put that into the script.
  - Try to write a simple script to bring down the application. Again, figure out what the administrator would do at the keyboard, then put that into the script.

## **Integrating HA Applications in Multiple Systems**

1. Install the application on a second system.
  - Create the LVM infrastructure on the second system.
  - Add the appropriate users to the system.
  - Install the appropriate executables.
  - With the application *not* running on the first system, try to bring it up on the second system. You might use the script you created in the step above. Is there anything different that you must do? Does it run?
  - Repeat this process until you can get the application to run on the second system.
2. Configure the Serviceguard cluster:
  - Create the cluster configuration.
  - Create a package.
  - Create the package script.
  - Use the simple scripts you created in earlier steps as the customer defined functions in the package control script.
3. Start the cluster and verify that applications run as planned.

## Testing the Cluster

### 1. Test the cluster:

- Have clients connect.
- Provide a normal system load.
- Halt the package on the first node and move it to the second node:

```
# cmhaltpkg pkg1  
# cmrunpkg -n node2 pkg1  
# cmmodpkg -e pkg1
```

- Move it back.  

```
# cmhaltpkg pkg1  
# cmrunpkg -n node1 pkg1  
# cmmodpkg -e pkg1
```
- Fail one of the systems. For example, turn off the power on node 1. Make sure the package starts up on node 2.
- Repeat failover from node 2 back to node 1.

### 2. Be sure to test all combinations of application load during the testing. Repeat the failover processes under different application states such as heavy user load versus no user load, batch jobs vs online transactions, etc.

### 3. Record timelines of the amount of time spent during the failover for each application state. A sample timeline might be 45 seconds to reconfigure the cluster, 15 seconds to run `fsck` on the filesystems, 30 seconds to start the application and 3 minutes to recover the database.

# D **Blank Planning Worksheets**

This appendix reprints blank versions of the planning worksheets described in the “Planning” chapter. You can duplicate any of these worksheets that you find useful and fill them in as a part of the planning process. The worksheets included in this appendix are as follows:

- Hardware Worksheet
- Power Supply Worksheet
- Quorum Server Worksheet
- Volume Group and Physical Volume Worksheet
- Cluster Configuration Worksheet
- Package Configuration Worksheet
- Package Control Script Worksheet

---

## Hardware Worksheet

=====

SPU Information:

Host Name \_\_\_\_\_ Server Series \_\_\_\_\_

Memory Capacity \_\_\_\_\_ Number of I/O Slots \_\_\_\_\_

=====

LAN Information:

Name of Master _____	Name of Interface _____	Node IP Addr _____	Traffic Type _____
-------------------------	----------------------------	-----------------------	-----------------------

Name of Master _____	Name of Interface _____	Node IP Addr _____	Traffic Type _____
-------------------------	----------------------------	-----------------------	-----------------------

Name of Master _____	Name of Interface _____	Node IP Addr _____	Traffic Type _____
-------------------------	----------------------------	-----------------------	-----------------------

=====

Quorum Server Name: \_\_\_\_\_ IP Address: \_\_\_\_\_

=====

Disk I/O Information for Shared Disks:

Bus Type \_\_\_\_\_ Slot Number \_\_\_\_\_ Address \_\_\_\_\_ Disk Device File \_\_\_\_\_

Bus Type \_\_\_\_\_ Slot Number \_\_\_\_\_ Address \_\_\_\_\_ Disk Device File \_\_\_\_\_

Bus Type \_\_\_\_\_ Slot Number \_\_\_\_\_ Address \_\_\_\_\_ Disk Device File \_\_\_\_\_

Bus Type \_\_\_\_\_ Slot Number \_\_\_\_\_ Address \_\_\_\_\_ Disk Device File \_\_\_\_\_

Power Supply Worksheet

SPU Power:

Host Name \_\_\_\_\_ Power Supply \_\_\_\_\_

Host Name \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Power:

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Disk Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Tape Backup Power:

Tape Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Tape Unit \_\_\_\_\_ Power Supply \_\_\_\_\_

Other Power:

Unit Name \_\_\_\_\_ Power Supply \_\_\_\_\_

Unit Name \_\_\_\_\_ Power Supply \_\_\_\_\_

---

## Quorum Server Worksheet

Quorum Server Data:

=====

QS Hostname: \_\_\_\_\_ IP Address: \_\_\_\_\_

OR

Cluster Name: \_\_\_\_\_

Package Name: \_\_\_\_\_ Package IP Address: \_\_\_\_\_

Hostname Given to Package by Network Administrator: \_\_\_\_\_

=====

Quorum Services are Provided for:

Cluster Name: \_\_\_\_\_

Host Names \_\_\_\_\_

Host Names \_\_\_\_\_

Cluster Name: \_\_\_\_\_

Host Names \_\_\_\_\_

Host Names \_\_\_\_\_

Cluster Name: \_\_\_\_\_

Host Names \_\_\_\_\_

Host Names \_\_\_\_\_



---

## Volume Group and Physical Volume Worksheet

=====

Volume Group Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

=====

Volume Group Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

Physical Volume Name: \_\_\_\_\_

---

## Cluster Configuration Worksheet

=====  
Name and Nodes:  
=====

Cluster Name: \_\_\_\_\_

Node Names: \_\_\_\_\_

Maximum Configured Packages: \_\_\_\_\_  
=====

Cluster Lock Data:  
=====

If using a quorum server:

Quorum Server Host Name or IP Address: \_\_\_\_\_

Quorum Server Polling Interval: \_\_\_\_\_ microseconds

Quorum Server Timeout Extension: \_\_\_\_\_ microseconds  
=====

If using a lock lun:

Lock LUN Name on Node 1: \_\_\_\_\_

Lock LUN Name on Node 2: \_\_\_\_\_

Lock LUN Name on Node 3: \_\_\_\_\_

Lock LUN Name on Node 4: \_\_\_\_\_  
=====

Subnets:  
=====

Heartbeat Subnet: \_\_\_\_\_

Monitored Non-heartbeat Subnet: \_\_\_\_\_

Monitored Non-heartbeat Subnet: \_\_\_\_\_  
=====

Timing Parameters:  
=====

Heartbeat Interval: \_\_\_\_\_

Node Timeout: \_\_\_\_\_

Network Polling Interval: \_\_\_\_\_  
=====

Autostart Delay: \_\_\_\_\_

Access Policies

User: \_\_\_\_\_

Host: \_\_\_\_\_

Role: \_\_\_\_\_

User: \_\_\_\_\_

Host: \_\_\_\_\_

Role: \_\_\_\_\_

---

## Package Configuration Worksheet

=====  
Package Configuration File Data:  
=====

Package Name: \_\_\_\_\_

Failover Policy: \_\_\_\_\_ Failback Policy: \_\_\_\_\_

Primary Node: \_\_\_\_\_

First Failover Node: \_\_\_\_\_

Additional Failover Nodes: \_\_\_\_\_

Package Run Script: \_\_\_\_\_ Timeout: \_\_\_\_\_

Package Halt Script: \_\_\_\_\_ Timeout: \_\_\_\_\_

Package AutoRun Enabled? \_\_\_\_\_

Node Failfast Enabled? \_\_\_\_\_

### Access Policies

User: \_\_\_\_\_

Host: \_\_\_\_\_

Role: \_\_\_\_\_

User: \_\_\_\_\_

Host: \_\_\_\_\_

Role: \_\_\_\_\_

---

## Package Control Script Worksheet

PACKAGE CONTROL SCRIPT WORKSHEET

Page \_\_\_\_ of \_\_\_\_

=====

Package Control Script Data:

=====

PATH\_\_\_\_\_

VGCHANGE\_\_\_\_\_

VG[0]\_\_\_\_\_LV[0]\_\_\_\_\_FS[0]\_\_\_\_\_

VG[1]\_\_\_\_\_LV[1]\_\_\_\_\_FS[1]\_\_\_\_\_

VG[2]\_\_\_\_\_LV[2]\_\_\_\_\_FS[2]\_\_\_\_\_

FS Umount Count: \_\_\_\_\_FS Mount Retry Count:\_\_\_\_\_

IP[0] \_\_\_\_\_ SUBNET \_\_\_\_\_

IP[1] \_\_\_\_\_ SUBNET \_\_\_\_\_

Service Name: \_\_\_\_\_ Command: \_\_\_\_\_ Restart: \_\_\_\_\_

Service Name: \_\_\_\_\_ Command: \_\_\_\_\_ Restart: \_\_\_\_\_

---

### NOTE

MD, RAIDTAB, and RAIDSTART are deprecated and should not be used. See the “Multipath for Storage” section of the current release notes for information on implementing multipath in a Serviceguard for Linux cluster.

---



**A**

- active node, 20
- adding a package to a running cluster, 256
- adding nodes to a running cluster, 244
- adding packages on a running cluster, 203
- administration
  - adding nodes to a running cluster, 244
  - cluster and package states, 213
  - halting a package, 252
  - halting the entire cluster, 246
  - moving a package, 253
  - of packages and services, 251
  - of the cluster, 242
  - reconfiguring a package while the cluster is running, 255
  - reconfiguring a package with the cluster offline, 254
  - reconfiguring the cluster, 247
  - removing nodes from operation in a running cluster, 245
  - responding to cluster events, 260
  - reviewing configuration files, 277
  - starting a package, 251
  - troubleshooting, 273
- adoptive node, 20
- applications
  - automating, 298
  - checklist of steps for integrating with Serviceguard, 321
  - handling failures, 316
  - writing HA services for networks, 299
- ARP messages
  - after switching, 74
- ASCII cluster configuration file template, 162
- ASCII package configuration file template, 182
- AUTO\_RUN
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 110
- AUTO\_START\_TIMEOUT
  - parameter in the cluster configuration file, 104
- AUTO\_START\_TIMEOUT (autostart delay)
  - in sample configuration file, 165
  - parameter in cluster manager configuration, 104
- automatic fallback
  - configuring with failover policies, 52
- automatic restart of cluster, 40

- automatic switching
  - parameter in package configuration, 110
- automatically restarting the cluster, 247
- automating application operation, 298
- autostart delay
  - parameter in the cluster configuration file, 104
- autostart for clusters
  - setting up, 175

**B**

- binding
  - in network applications, 310
- bridged net
  - defined, 27
- building a cluster
  - ASCII cluster configuration file template, 162
  - logical volume infrastructure, 147
  - verifying the cluster configuration, 170
- bus type
  - hardware planning, 93

**C**

- changes in cluster membership, 40
- changes to cluster allowed while the cluster is running, 248
- changes to packages allowed while the cluster is running, 258
- checkpoints, 303
- client connections
  - restoring in applications, 314
- cluster
  - configuring with commands, 162
  - redundancy of components, 26
  - Serviceguard, 18
  - typical configuration, 18
  - understanding components, 26
- cluster administration, 242
  - solving problems, 279
- cluster and package maintenance, 211
- cluster configuration
  - file on all nodes, 38
  - planning, 99
  - planning worksheet, 106
  - sample diagram, 89
  - verifying the cluster configuration, 170
- cluster configuration file, 165
  - Autostart Delay parameter (AUTO\_START\_TIMEOUT), 104

---

# Index

- cluster coordinator
    - defined, 38
  - cluster lock
    - 4 or more nodes, 43
    - and power supplies, 30
    - storing configuration data, 160
    - two nodes, 41, 42
    - use in re-forming a cluster, 41, 42
  - cluster manager
    - automatic restart of cluster, 40
    - blank planning worksheet, 330
    - cluster node parameter, 101, 102
    - defined, 38
    - dynamic re-formation, 40
    - heartbeat interval parameter, 103
    - heartbeat subnet parameter, 102
    - initial configuration of the cluster, 38
    - main functions, 38
    - maximum configured packages parameter, 105
    - monitored non-heartbeat subnet, 103
    - network polling interval parameter, 105
    - node timeout parameter, 104
    - planning the configuration, 101
    - testing, 265
  - cluster node
    - parameter in cluster manager configuration, 101, 102
  - cluster parameters
    - initial configuration, 38
  - cluster startup
    - manual, 39
  - CLUSTER\_NAME (cluster name)
    - in sample configuration file, 165
  - cmapplyconf, 247
  - cmcheckconf, 170, 204
    - troubleshooting, 278
  - cmdeleteconf
    - deleting a package configuration, 256
    - deleting the cluster configuration, 177
  - cmhaltnode, 213
  - cmmodnet
    - assigning IP addresses in control scripts, 68
  - cmmodpkg, 215
  - cmquerycl
    - troubleshooting, 278
  - cmscancl, 273
  - cmviewcl, 216
  - CONCURRENT\_FSCK\_OPERATIONS
    - parameter in package control script, 116
  - CONCURRENT\_MOUNT\_OPERATIONS
    - parameter in package control script, 116
  - configuration
    - ASCII cluster configuration file template, 162
    - ASCII package configuration file template, 182
    - basic tasks and steps, 23
    - cluster planning, 99
      - of the cluster, 38
    - package, 179
    - package planning, 107
    - service, 179
  - configuration file
    - for cluster manager, 38
    - troubleshooting, 277
  - configuring packages and their services, 179
  - control script
    - adding customer defined functions, 202
    - creating with commands, 188
    - in package configuration, 188
    - pathname parameter in package configuration, 111
    - troubleshooting, 277
  - controlling the speed of application failover, 300
  - creating the package configuration, 181
  - customer defined functions
    - adding to the control script, 202
- ## D
- data
    - disks, 28
  - data congestion, 39
  - deciding when and where to run packages, 46
  - deleting a package configuration
    - using cmdeleteconf, 256
  - deleting a package from a running cluster, 256
  - deleting the cluster configuration
    - using cmdeleteconf, 177
  - designing applications to run on multiple systems, 305
  - disk
    - data, 28
    - interfaces, 28
    - root, 28
    - sample configurations, 29
  - disk I/O
    - hardware planning, 93



- disk layout
  - planning, 98
- disk logical units
  - hardware planning, 93
- disks
  - in Serviceguard, 28
  - replacing, 268
  - supported types in Serviceguard, 28
- distributing the cluster and package configuration, 204, 205
- down time
  - minimizing planned, 318
- dynamic cluster re-formation, 40
- E**
- enclosure for disks
  - replacing a faulty mechanism, 268
- Ethernet
  - redundant configuration, 27
- expanding the cluster
  - planning ahead, 88
- expansion
  - planning for, 109
- F**
- failback policy
  - package configuration file parameter, 110
  - used by package manager, 52
- FAILBACK\_POLICY parameter
  - in package configuration file, 110
  - used by package manager, 52
- failover
  - controlling the speed in applications, 300
  - defined, 20
- failover behavior
  - in packages, 55
- failover policy
  - package configuration parameter, 109
  - used by package manager, 49
- FAILOVER\_POLICY parameter
  - in package configuration file, 109
  - used by package manager, 49
- failure
  - kinds of responses, 81
  - network communication, 83
  - response to hardware failures, 82
  - responses to package and service failures, 83
  - restarting a service after failure, 83
- failures

- of applications, 316
- FibreChannel, 28
- figures
  - mirrored disks connected for high availability, 29
  - redundant LANs, 27
  - sample cluster configuration, 89
  - Serviceguard software components, 32
  - tasks in configuring an Serviceguard cluster, 23
  - typical cluster after failover, 20
  - typical cluster configuration, 18
- file locking, 313
- file system
  - in control script, 114
- file system name parameter in package control script, 115
- file system type parameter in package control script, 115
- file systems
  - planning, 98
- Filesystem mount retry count, 116
- Filesystem unmount count, 115, 116
- floating IP address
  - defined, 68
- floating IP addresses
  - in Serviceguard packages, 68
- FS, 115
  - in sample package control script, 191
- FS\_MOUNT\_OPT
  - in sample package control script, 191
- FS\_TYPE, 115
- G**
- general planning, 87
- gethostbyname
  - and package IP addresses, 68
- gethostbyname(), 308
- H**
- HA NFS server parameter in package control script, 117
- HA\_NFS\_SERVER, 117
- HALT\_SCRIPT
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 111
- HALT\_SCRIPT\_TIMEOUT (halt script timeout)

---

# Index

- in sample ASCII package configuration file, 182
- parameter in package configuration, 112
- halting a cluster, 246
- halting a package, 252
- halting the entire cluster, 246
- handling application failures, 316
- hardware
  - monitoring, 267
  - power supplies, 30
- hardware failures
  - response to, 82
- hardware planning
  - blank planning worksheet, 325
  - Disk I/O Bus Type, 93
  - disk I/O information for shared disks, 93
  - host IP address, 90, 97
  - host name, 90
  - I/O bus addresses, 93
  - I/O slot numbers, 93
  - LAN interface name, 90, 97
  - LAN traffic type, 91
  - memory capacity, 90
  - number of I/O slots, 90
  - planning the configuration, 89
  - S800 series number, 90
  - SPU information, 90
  - subnet, 90, 97
  - worksheet, 95
- heartbeat interval
  - parameter in cluster manager configuration, 103
- heartbeat messages, 19
  - defined, 38
- heartbeat subnet address
  - parameter in cluster manager configuration, 102
- HEARTBEAT\_INTERVAL
  - in sample configuration file, 165
- HEARTBEAT\_INTERVAL (heartbeat timeout)
  - parameter in cluster manager configuration, 103
- HEARTBEAT\_IP
  - in sample configuration file, 165
  - parameter in cluster manager configuration, 102
- high availability, 18
  - HA cluster defined, 26
  - objectives in planning, 87

- host IP address
  - hardware planning, 90, 97
- host name
  - hardware planning, 90
- how the cluster manager works, 38
- how the network manager works, 68

## I

- I/O bus addresses
  - hardware planning, 93
- I/O slots
  - hardware planning, 90, 93
- installing software
  - Quorum Server, 133
- integrating HA applications with Serviceguard, 321
- introduction
  - Serviceguard at a glance, 18
  - understanding Serviceguard hardware, 25
  - understanding Serviceguard software, 31
- IP
  - in sample package control script, 191
  - IP address array variable in package control script, 116
- IP address
  - adding and deleting in packages, 69
  - for nodes and packages, 68
  - hardware planning, 90, 97
  - portable, 68
  - reviewing for packages, 274
  - switching, 47, 48, 74

## J

- JFS, 301

## K

- kernel consistency
  - in cluster configuration, 128, 129

## L

- LAN
  - heartbeat, 38
  - interface name, 90, 97
- LAN failure
  - Serviceguard behavior, 26
- LAN interfaces
  - primary and secondary, 27
- LAN planning
  - host IP address, 90, 97

- traffic type, 91
- link-level addresses, 307
- load sharing with IP addresses, 69
- local switching, 70
- lock
  - cluster locks and power supplies, 30
  - use of the cluster lock, 42
  - use of the cluster lock disk, 41
- lock volume group, reconfiguring, 247
- logical volume parameter in package control script, 115
- logical volumes
  - creating the infrastructure, 147
  - planning, 98
- LV, 115
  - in sample package control script, 191
- LVM
  - commands for cluster use, 147
  - creating file systems, 107
  - creating logical volumes, 107
  - creating volume groups, 107
  - disks, 28
  - planning, 98

## M

- MAC addresses, 307
- man pages
  - list of pages available for Serviceguard, 285
- managing the cluster and nodes, 242
- manual cluster startup, 39
- MAX\_CONFIGURED\_PACKAGES
  - parameter in cluster manager configuration, 105
- maximum number of nodes, 26
- MD, 115
  - in sample package control script, 191
- membership change
  - reasons for, 40
- memory capacity
  - hardware planning, 90
- memory requirements
  - lockable memory for Serviceguard, 88
- minimizing planned down time, 318
- mirrored disks connected for high availability
  - figure, 29
- monitor cluster with Serviceguard
  - commands, 172
- monitored non-heartbeat subnet
  - parameter in cluster manager configuration, 103

- monitored resource failure
  - Serviceguard behavior, 26
- monitoring hardware, 267
- mount options
  - in control script, 114
- moving a package, 253
- multiple device parameter in control script, 115
- multiple systems
  - designing applications for, 305

## N

- network
  - adding and deleting package IP addresses, 69
  - load sharing with IP addresses, 69
  - local interface switching, 70
  - redundancy, 27
  - remote system switching, 74
- network communication failure, 83
- network components
  - in Serviceguard, 27
- network manager
  - adding and deleting package IP addresses, 69
  - main functions, 68
- network planning
  - subnet, 90, 97
- network polling interval
  - (NETWORK\_POLLING\_INTERVAL)
    - parameter in cluster manager configuration, 105
- NETWORK\_INTERFACE
  - in sample configuration file, 165
- NETWORK\_POLLING\_INTERVAL
  - (network polling interval)
    - in sample configuration file, 165
- networking
  - redundant subnets, 90
- networks
  - binding to IP addresses, 310
  - binding to port addresses, 310
  - IP addresses and naming, 305
  - node and package IP addresses, 68
  - packages using IP addresses, 308
  - supported types in Serviceguard, 27
  - writing network applications as HA services, 299
- node
  - basic concepts, 26

---

# Index

- in Serviceguard cluster, 18
- IP addresses, 68
- node types
  - active, 20
  - primary, 20
- NODE\_FAIL\_FAST\_ENABLED
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 111
- NODE\_NAME
  - in sample ASCII package configuration file, 182
  - parameter in cluster manager configuration, 101, 102
- NODE\_TIMEOUT (heartbeat timeout)
  - in sample configuration file, 165
- NODE\_TIMEOUT (node timeout)
  - parameter in cluster manager configuration, 104
- nodetypes
  - primary, 20

## O

- Object Manager, 277
- optimizing packages for large numbers of storage units, 190
- outages
  - insulating users from, 298

## P

- package
  - adding and deleting package IP addresses, 69
  - basic concepts, 26
  - blank planning worksheet, 332, 333
  - changes allowed while the cluster is running, 258
  - halting, 252
  - in Serviceguard cluster, 18
  - local interface switching, 70
  - moving, 253
  - reconfiguring while the cluster is running, 255
  - reconfiguring with the cluster offline, 254
  - remote switching, 74
  - starting, 251
- package administration, 251
  - solving problems, 279
- package and cluster maintenance, 211

- package configuration
  - automatic switching parameter, 110
  - control script pathname parameter, 110
  - distributing the configuration file, 204, 205
  - failback policy parameter, 110
  - failover policy parameter, 109
  - package failfast parameter, 111
  - package name parameter, 109
  - planning, 107
  - run and halt script timeout parameters, 112
  - service fail fast parameter, 113
  - service halt timeout parameter, 113
  - service name parameter, 113
  - step by step, 179
  - subnet parameter, 113
  - using commands, 181
  - verifying the configuration, 204, 205
  - writing the package control script, 188
- package configuration file, 182
- package control script
  - CONCURRENT\_FSCK\_OPERATIONS
    - parameter, 116
  - CONCURRENT\_MOUNT\_OPERATIONS
    - parameter, 116
  - FS parameter, 115
  - FS\_MOUNT\_RETRY\_COUNT, 116
  - FS\_TYPE parameter, 115
  - FS\_UMOUNT\_COUNT, 115, 116
  - generating with commands, 188
  - HA\_NFS\_SERVER parameter, 117
  - IP addresses, 116
  - LV parameter, 115
  - MD parameter, 115
  - SERVICE\_CMD parameter, 117
  - SERVICE\_NAME parameter, 117
  - SERVICE\_RESTART parameter, 118
  - subnets, 116
  - worksheet, 118
- package coordinator
  - defined, 38
- package failfast
  - parameter in package configuration, 111
- package failover behavior, 55
- package failures
  - responses, 83
- package IP address
  - defined, 68
- package IP addresses
  - defined, 68
  - reviewing, 274

- package manager
    - blank planning worksheet, 332, 333
    - testing, 264
  - package name
    - parameter in package configuration, 109
  - package switching behavior
    - changing, 257
  - PACKAGE\_NAME
    - in sample ASCII package configuration file, 182
  - packages
    - deciding where and when to run, 46
  - parameters
    - for failover, 55
  - parameters for cluster manager
    - initial configuration, 38
  - PATH, 114
  - performance
    - optimizing packages for large numbers of storage units, 190
  - performance variables in package control script, 116
  - physical volume
    - for cluster lock, 41, 42
    - parameter in cluster lock configuration, 101
  - physical volumes
    - blank planning worksheet, 329
    - planning, 98
  - planning
    - cluster configuration, 99
    - cluster manager configuration, 101
    - disk I/O information, 93
    - for expansion, 109
    - hardware configuration, 89
    - high availability objectives, 87
    - overview, 85
    - package configuration, 107
    - power, 96
    - SPU information, 90
    - volume groups and physical volumes, 98
    - worksheets, 95
  - planning and documenting an HA cluster, 85
  - planning for cluster expansion, 88
  - planning worksheets
    - blanks, 325
  - point of failure
    - in networking, 27
  - ports
    - dual and single aggregated, 71
  - power planning
    - power sources, 96
    - worksheet, 96, 97, 328
  - power supplies
    - blank planning worksheet, 327
  - power supply
    - and cluster lock, 30
    - UPS, 30
  - primary LAN interfaces
    - defined, 27
  - primary node, 20
- ## Q
- Quorum Server
    - installing, 133, 134
    - running, 135
    - running in a package, 136
  - quorum server
    - planning, 100, 101
- ## R
- RAIDSTART
    - in sample package control script, 191
  - RAIDSTOP
    - in sample package control script, 191
  - RAIDTAB
    - in sample package control script, 191
  - reconfiguring a package
    - while the cluster is running, 255
  - reconfiguring a package with the cluster
    - offline, 254
  - reconfiguring a running cluster, 248
  - reconfiguring the entire cluster, 247
  - reconfiguring the lock volume group, 247
  - recovery time, 99
  - redundancy
    - in networking, 27
    - of cluster components, 26
  - redundancy in network interfaces, 27
  - redundant Ethernet configuration, 27
  - redundant LANS
    - figure, 27
  - redundant networks
    - for heartbeat, 19
  - re-formation
    - of cluster, 40
  - relocatable IP address
    - defined, 68
  - relocatable IP addresses
    - in Serviceguard packages, 68
  - remote switching, 74
-

---

# Index

- removing nodes from operation in a running cluster, 245
- removing packages on a running cluster, 203
- removing Serviceguard from a system, 262
- replacing disks, 268
- resources
  - disks, 28
- responses
  - to cluster events, 260
  - to package and service failures, 83
- responses to failures, 81
- responses to hardware failures, 82
- restart
  - automatic restart of cluster, 40
  - following failure, 83
  - SERVICE\_RESTART variable in package control script, 118
- restartable transactions, 302
- restarting the cluster automatically, 247
- restoring client connections in applications, 314
- retry count, 116
- rhosts file
  - for security, 124
- rotating standby
  - configuring with failover policies, 50
  - setting package policies, 50
- RUN\_SCRIPT
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 111
- RUN\_SCRIPT\_TIMEOUT
  - in sample ASCII package configuration file, 182
- RUN\_SCRIPT\_TIMEOUT (run script timeout)
  - parameter in package configuration, 112
- running cluster
  - adding or removing packages, 203
- S**
- S800 series number
  - hardware planning, 90
- sample cluster configuration
  - figure, 89
- sample disk configurations, 29
- SCSI addressing, 100, 101
- security
  - editing files, 124
- service administration, 251
- service command
  - variable in package control script, 117
- service configuration
  - step by step, 179
- service fail fast
  - parameter in package configuration, 113
- service failures
  - responses, 83
- service halt timeout
  - parameter in package configuration, 113
- service name
  - parameter in package configuration, 113
  - variable in package control script, 117
- service restarts, 83
- SERVICE\_CMD
  - array variable in package control script, 117
  - in sample package control script, 191
- SERVICE\_FAIL\_FAST\_ENABLED
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 113
- SERVICE\_HALT\_TIMEOUT
  - in sample ASCII package configuration file, 182
  - parameter in package configuration, 113
- SERVICE\_NAME
  - array variable in package control script, 117
  - in sample ASCII package configuration file, 182
  - in sample package control script, 191
  - parameter in package configuration, 113
- SERVICE\_RESTART
  - array variable in package control script, 118
  - in sample package control script, 191
- SERVICE\_RESTART parameter
  - variable in package control script, 118
- Serviceguard
  - introduction, 18
- Serviceguard at a Glance, 17
- Serviceguard behavior
  - in LAN failure, 26
  - in monitored resource failure, 26
  - in software failure, 26
- Serviceguard commands
  - using to configure package, 181
- Serviceguard Manager
  - managing cluster objects, 236
  - overview, 224
  - viewing cluster objects, 232
- Serviceguard software components
  - figure, 32

- shared disks
  - planning, 93
- shutdown and startup
  - defined for applications, 299
- single point of failure
  - avoiding, 18
- single-node operation, 176, 261
- SNA applications, 312
- software failure
  - Serviceguard behavior, 26
- software planning
  - LVM, 98
- solving problems, 279
- SPU information
  - planning, 90
- standby LAN interfaces
  - defined, 27
- starting a package, 251
- startup and shutdown
  - defined for applications, 299
- startup of cluster
  - manual, 39
- state
  - of cluster and package, 213
- stationary IP addresses, 68
- STATIONARY\_IP
  - parameter in cluster manager
    - configuration, 103
- status
  - of cluster and package, 213
  - package IP address, 274
  - system log file, 275
- stopping a cluster, 246
- SUBNET
  - array variable in package control script, 116
  - in sample ASCII package configuration file, 182
  - in sample package control script, 191
  - parameter in package configuration, 113
- subnet
  - hardware planning, 90, 97
  - parameter in package configuration, 113
- supported disks in Serviceguard, 28
- supported networks in Serviceguard, 27
- switching
  - ARP messages after switching, 74
  - local interface switching, 70
  - remote system switching, 74
- switching IP addresses, 47, 48, 74
- system log, 267
- system log file
  - troubleshooting, 275
- system message
  - changing for clusters, 176
- T**
- tasks in Serviceguard configuration
  - figure, 23
- template
  - ASCII cluster configuration file, 162
  - ASCII package configuration file, 182
- testing
  - cluster manager, 265
  - package manager, 264
- testing cluster operation, 264
- TOC
  - when a node fails, 81
- traffic type
  - LAN hardware planning, 91
- troubleshooting
  - approaches, 273
  - monitoring hardware, 267
  - replacing disks, 268
  - reviewing control scripts, 277
  - reviewing package IP addresses, 274
  - reviewing system log file, 275
  - using cmquerycl and cmcheckconf, 278
- troubleshooting your cluster, 263
- typical cluster after failover
  - figure, 20
- typical cluster configuration
  - figure, 18
- U**
- uname(2), 309
- understanding network components in Serviceguard, 27
- unmount count, 115, 116
- UPS
  - in power planning, 96
  - power supply, 30
- use of the cluster lock, 41, 42
- V**
- verifying cluster configuration, 170
- verifying the cluster and package configuration, 204, 205
- VG
  - in sample package control script, 191
- vgcfgbackup
  - and cluster lock data, 160

---

# Index

VGChange, 114  
volume group  
    for cluster lock, 41, 42  
    planning, 98  
volume group and physical volume planning,  
    98  
Volume groups  
    in control script, 114

## W

What is Serviceguard?, 18  
worksheet  
    blanks, 325  
    cluster configuration, 106, 331  
    hardware configuration, 95, 325  
    package configuration, 332, 333  
    package control script, 118  
    power supply configuration, 96, 97, 327, 328  
    use in planning, 85