# Managing Serviceguard Version A.11.16, Eleventh Edition Second Printing


**Manufacturing Part Number : B3936-90079**

**May 2004**
**reprinted June 2005**

# Legal Notices



Corporate Offices:

*Hewlett-Packard Co.*
*3000 Hanover St.*
*Palo Alto, CA 94304*



Printed in the US.

# Contents

# Contents

# Contents

# Contents

# Contents

# Contents

# Contents

# Tables

# Tables

# Figures

# Figures

# Printing History

**Table 1**

| Printing Date | Part Number | Edition |
|---|---|---|
| January 1995 | B3936-90001 | First |
| June 1995 | B3936-90003 | Second |
| December 1995 | B3936-90005 | Third |
| August 1997 | B3936-90019 | Fourth |
| January 1998 | B3936-90024 | Fifth |
| October 1998 | B3936-90026 | Sixth |
| December 2000 | B3936-90045 | Seventh |
| September 2001 | B3936-90053 | Eighth |
| March 2002 | B3936-90065 | Ninth |
| June 2003 | B3936-90070 | Tenth |
| June 2004 | B3936-90076 | Eleventh |
| June 2004 (Reprint June 2005) | B3936-90076 | Eleventh, Second printing. Updates mainly to security and access |

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

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:

*Infrastructure Solutions Division*
*Hewlett-Packard Co.*
*19111 Pruneridge Ave.*
*Cupertino, CA 95014*

# Preface

This second printing adds new information to the first printing of the eleventh edition, primarily in “Editing Security Files” on page 182.

This guide describes how to configure Serviceguard to run on HP 9000 Series 800 or HP Integrity servers under the HP-UX operating system. The contents are as follows:

- Chapter 1, “Serviceguard at a Glance,” describes a Serviceguard cluster and provides a roadmap for using this guide.
- Chapter 2, “Understanding Serviceguard Hardware Configurations,” 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 HP-UX 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. This second printing of the eleventh edition includes new information in “Editing Security Files” on page 182.
- 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 and reprints summary information from each man page.
- Appendix B, “Enterprise Cluster Master Toolkit,” describes a group of tools available for creating specialized cluster configurations.

- Appendix C, “Designing Highly Available Cluster Applications,” gives guidelines for creating cluster-aware applications that provide optimal performance in an Serviceguard environment.
- Appendix D, “Integrating HA Applications with Serviceguard,” presents suggestions for integrating your existing applications with Serviceguard.
- Appendix E, “Rolling Software Upgrades,” shows how to move from one Serviceguard or HP-UX release to another without bringing down your applications.
- Appendix F, “Blank Planning Worksheets,” contains a set of empty worksheets for preparing a Serviceguard configuration.
- Appendix G, “Migrating from LVM to VxVM Data Storage,” describes how to convert from LVM data storage to VxVM data storage.
- Appendix H, “IPv6 Network Support,” describes the IPv6 addressing scheme and the primary/standby interface configurations supported.

**Related Publications**

The following documents contain additional useful information:

- From *http://www.docs.hp.com* High Availability:
  - *Clusters for High Availability: a Primer of HP Solutions*. Hewlett-Packard Professional Books: Prentice Hall PTR, 2001 (ISBN 0-13-089355-2)
  - *Managing HP Serviceguard for Linux, Fifth Edition*, (B9903-90046) May 2005
  - *Designing Disaster Tolerant High Availability Clusters* (B7660-90009)
  - *Configuring OPS Clusters with Serviceguard OPS Edition* (B5158-90044)
  - *HP Serviceguard Extension for Faster Failover, Version A.01.00, Release Notes*, June 2004 (T2389-90001)
  - *Managing Serviceguard Extension for SAP* December 2004 (T2357-90007)
  - *Enterprise Cluster Master Toolkit Version B.02.20 Release Notes* (HP-UX 11iv1) December 2004 (T1909-90019)

— *Enterprise Cluster Master Toolkit Version B.02.21 Release Notes* (HP-UX 11iv2) December 2004 (T1909-90024)

— *Using High Availability Monitors* (B5736-90042)

— *Using the Event Monitoring Service* (B7609-90022)

— *Managing Highly Available NFS* (B5140-90017)

— *HP Serviceguard Quorum Server Release Notes* (B8467-90026)

From *http://www.docs.hp.com* Networking and Communications:

— *HP Auto Port Aggregation (APA) Support Guide*

— *HP Auto Port Aggregation Release Notes* and other Auto Port Aggregation documents

From *http://www.docs.hp.com* HP-UX Operating Environments:

— *Managing Systems and Workgroups* (5990-8172)

From *http://www.docs.hp.com/en/hpux11.0.html*:

— *Using Advanced Tape Services* (B3936-90032)

Before attempting to use VxVM storage with Serviceguard, please refer to the following:

- VERITAS Volume Manager Administrator's Guide. This contains glossary of VERITAS terminology.
- VERITAS Volume Manager Storage Administrator Administrator's Guide
- VERITAS Volume Manager Reference Guide
- VERITAS Volume Manager Migration Guide
- VERITAS Volume Manager for HP-UX Release Notes

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

- `http://www.hp.com/go/ha`

Use the following URL for access to a wide variety of HP-UX documentation:

- `http://docs.hp.com`

**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 at a Glance

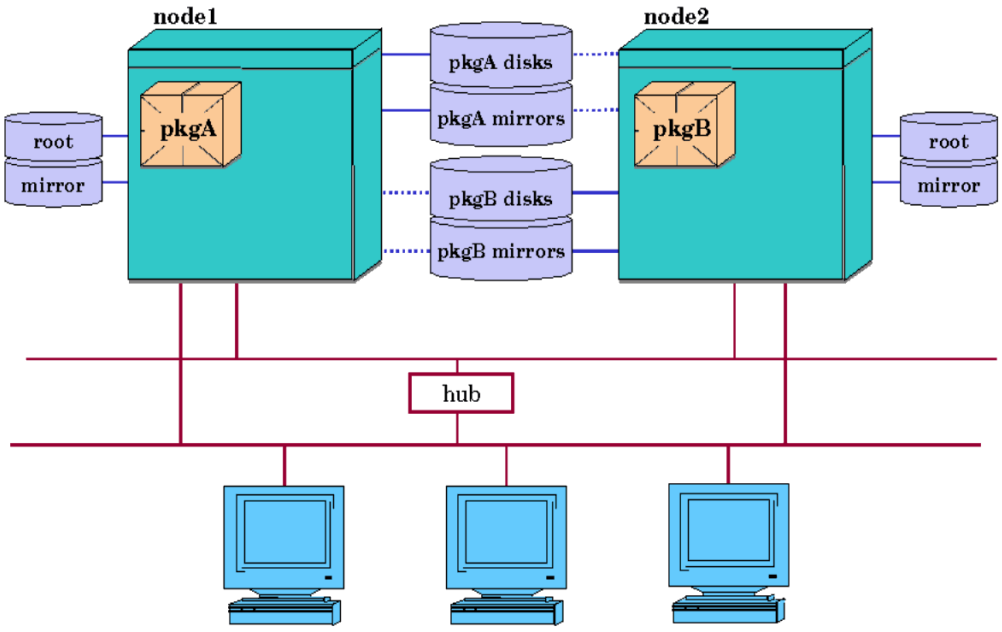
This chapter introduces Serviceguard on HP-UX, and shows where to find different kinds of information in this book. The following topics are presented:

- What is Serviceguard?
- Using Serviceguard Manager
- A Roadmap for Configuring Clusters and Packages

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.”

Figure 1-1 shows a typical Serviceguard cluster with two nodes.

**Figure 1-1 Typical Cluster Configuration**



# What is Serviceguard?

**Serviceguard** allows you to create high availability clusters of HP 9000 or 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 9000 or 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 HP-UX 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.

In Figure 1-1, 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 mirror copy of the data. Note that both nodes are physically connected to both groups of mirrored disks. 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).

Mirror copies of data 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.

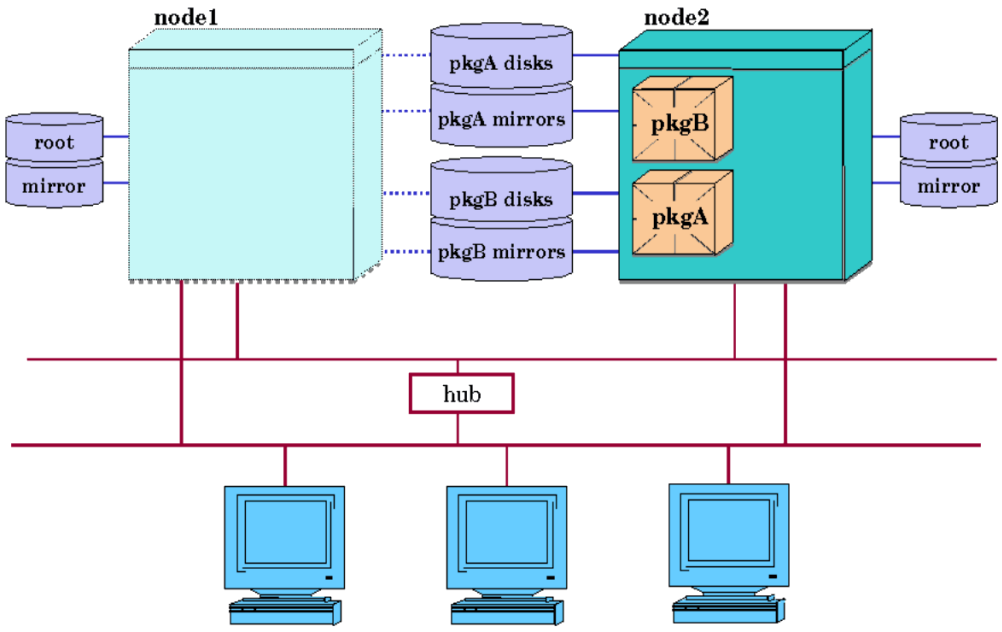
Note that the network hardware is cabled to provide redundant LAN interfaces on each node. Serviceguard uses TCP/IP network services for reliable communication among nodes in the cluster, including the transmission of **heartbeat messages**, signals from each functioning

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 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 MirrorDisk/UX or VERITAS Volume Manager, which provide disk redundancy to eliminate single points of failure in the disk subsystem; Event Monitoring Service (EMS), which lets you monitor and detect failures that are not directly handled by Serviceguard; 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.

# Using Serviceguard Manager

Serviceguard Manager is the graphical user interface for Serviceguard.

The Serviceguard Manager management station can be HP-UX, Linux, and Windows systems. From there, you can monitor, administer, and configure Serviceguard clusters on HP-UX or on Linux.

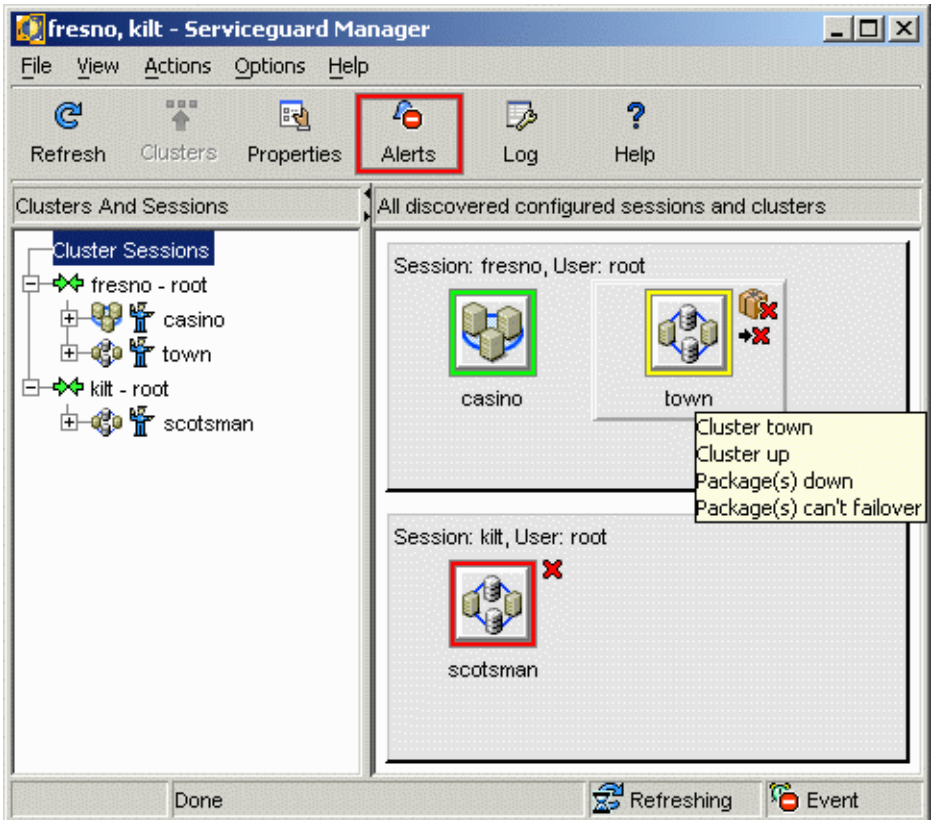
- Monitor: You can see information about Serviceguard objects on your subnets. The objects are represented in a hierarchal tree, in a graphical map. More information is available in the object's Properties.
- Administer: You can do administrative tasks through Serviceguard Manager, such as run or halt clusters and packages. These clusters must have Serviceguard Version A.11.12 and later.
- Configure: You can create or modify cluster and package configuration. These clusters must have Serviceguard Version A.11.16 and later.

Refer to the latest *Serviceguard Manager Release Notes* for additional information about Serviceguard Manager; they are posted at `http://docs.hp.com -> high availability ->` Serviceguard. Serviceguard Manager can be downloaded free from the Serviceguard Distributed Components Disk that comes with your Serviceguard disks, or from `http://software.hp.com`.

## Monitoring with Serviceguard Manager

From a management station, connect to a session server. You can request to see all the clusters the server can reach, or you can list specific clusters. You can also request to see all the unused nodes on the subnet - that is, all the nodes that are not currently configured in any cluster.

**Figure 1-3 Monitoring with Serviceguard Manager**

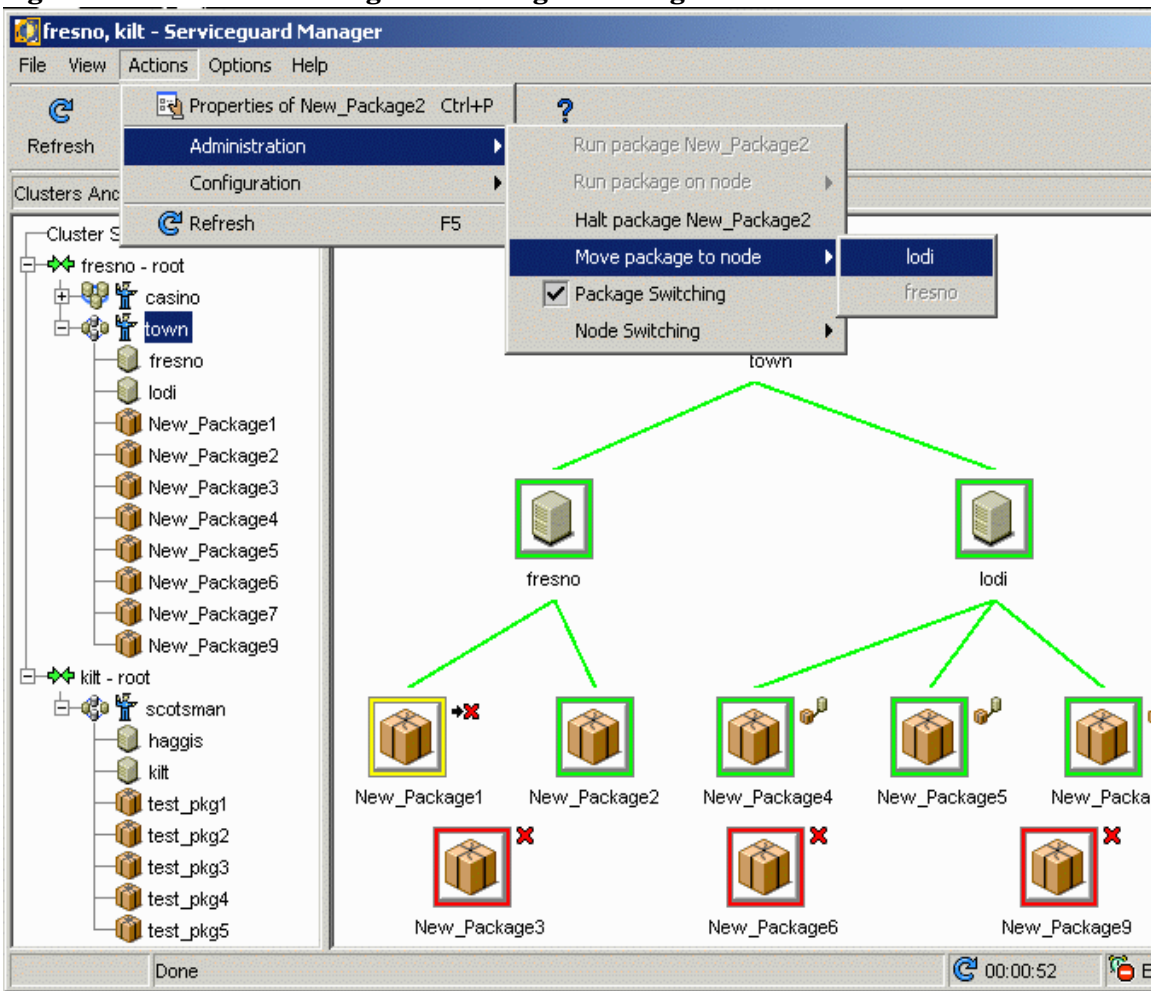



## Administering with Serviceguard Manager

You can also administer clusters, nodes, and packages if you have the appropriate access permissions (Serviceguard A.11.14 and A.11.15) or access control policies (Serviceguard A.11.16):

- Cluster: halt, run
- Cluster nodes: halt, run
- Package: halt, run, move from one node to another, reset node- and package-switching flags

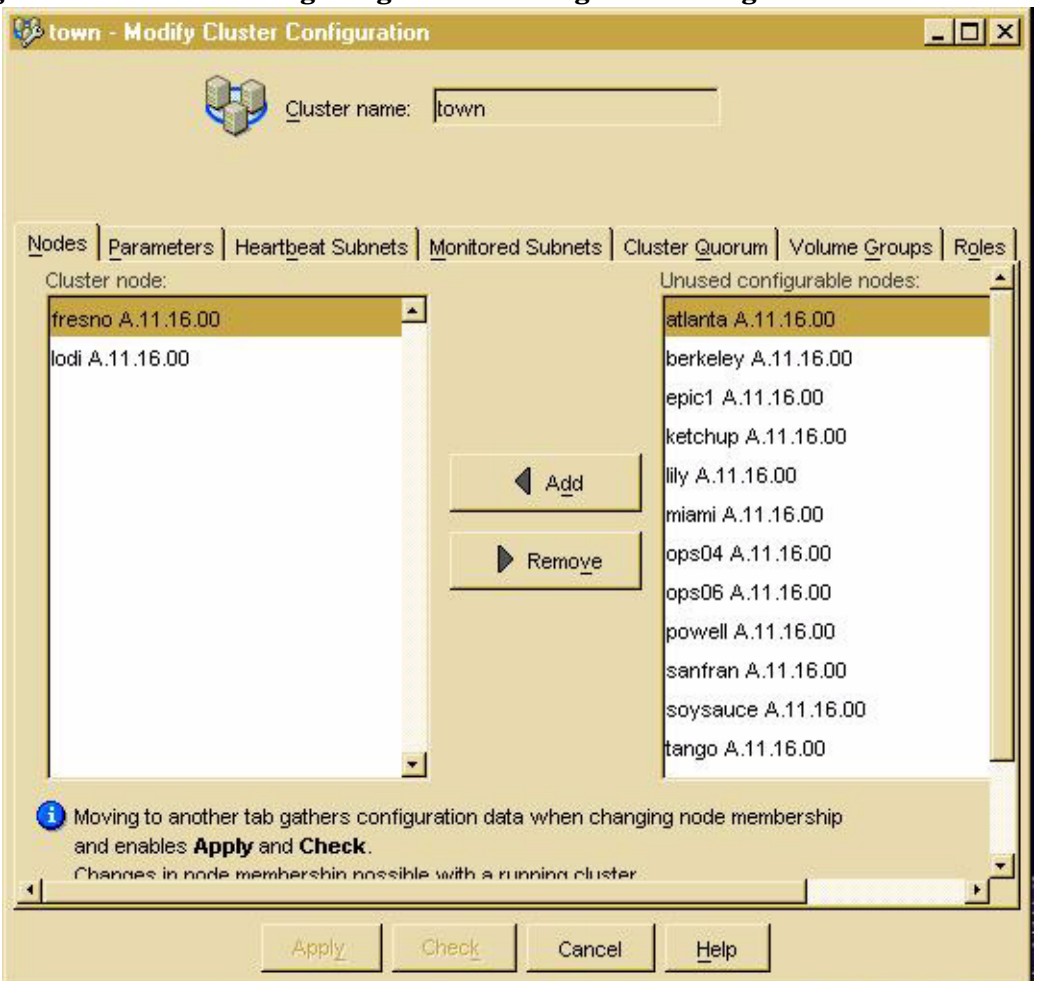
**Figure 1-4 Serviceguard Manager Package Administration**




## Configuring with Serviceguard Manager

With Serviceguard version A.11.16, you can also configure clusters and packages. Both the server node and the target cluster must have Serviceguard version A.11.16 installed, and you must have root (UID=0) login to the cluster nodes.

**Figure 1-5 Configuring with Serviceguard Manager**




## Serviceguard Manager Help

To see online help, click on the “Help” menu item at the top of the 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”

## How Serviceguard Manager Works

To start Serviceguard Manager on a Unix or Linux management station, type the `sgmgr` command. You can enter the options on the command line, or in a dialog box after the interface opens. For command syntax and options, check online Help -> Troubleshooting, or enter `man sgmgr` on the command line.

To start Serviceguard Manager on a Windows management station, double-click the icon on your desktop. To see or change the actual command used, right click the icon and choose Properties. See online Help -> Troubleshooting for command syntax and options.

To open a saved “snapshot” cluster file, specify a filename with the `.sgm` extension; you must have view permission on the file and its directory. viewing the example files is a good way to get acquainted with Serviceguard Manager.

To see “live” clusters, connect to a Serviceguard node’s Cluster Object Manager (COM) daemon. (The COM is automatically installed with Serviceguard.) This node becomes the session server. It goes out over its subnets, and establishes connections with the COMs on other Serviceguard nodes. The session server relays commands from the management station to the target nodes, and relays the target nodes’ configuration and status data back to the management station. It also relays operation messages from the target nodes to the management station.

To connect, you need to specify a valid username and password from the session server’s `/etc/passwd` file. List the cluster or clusters you want to see. Click “unused nodes” to see nodes that are not currently configured into a cluster, but have Serviceguard installed.

For the session server to get information from a cluster, the target cluster must allow it access. The target node will resolve the session server’s hostname through `/etc/hosts` or DNS. Access method and non-root roles changed in Serviceguard Version A.11.16:

- In Serviceguard version A.11.16 clusters, the cluster configuration file, or a package configuration file, must have an Access Control Policy with this triplet: the intended user, the COM server hostname, and a role of at least Monitor.
- In earlier versions of Serviceguard, the `/etc/cmcluster/cmclnodelist/` file must have this pair listed: COM host_node, and user root. For more information about earlier versions of Serviceguard, see the appropriate manual and release notes, posted at: `http://docs.hp.com/hpux/ha`.

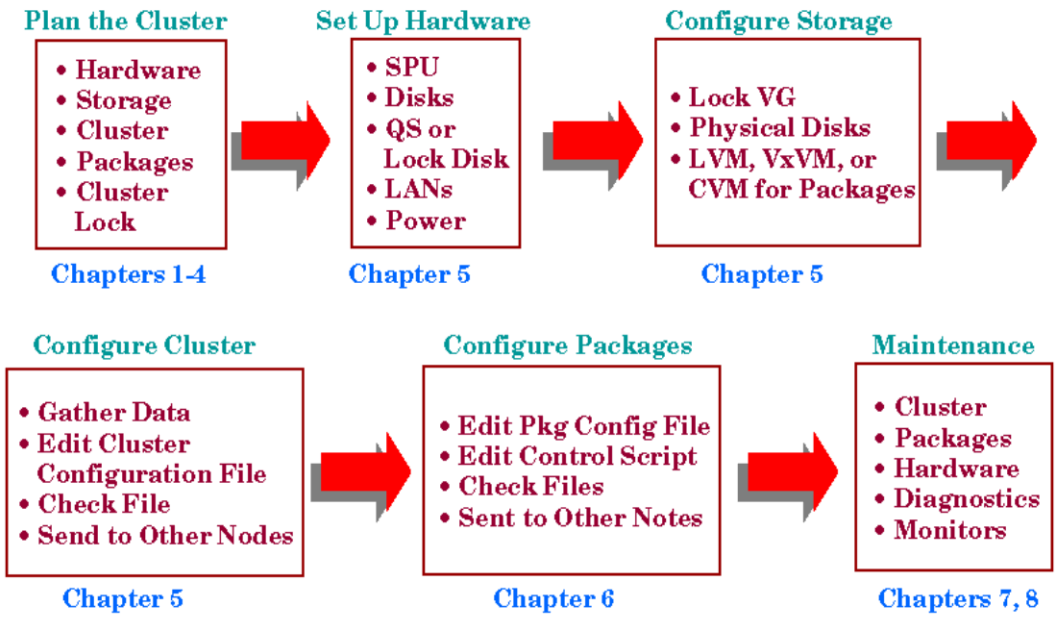
For information on access policies, refer to “Editing Security Files” on page 182.

Using Serviceguard Manager, you can create or modify cluster and package configuration for clusters with Serviceguard Version A.11.16, if you give the root (UID=0) password for a cluster’s node.

# A Roadmap for Configuring Clusters and Packages

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 1-6.

**Figure 1-6 Tasks in Configuring an Serviceguard Cluster**




The tasks in Figure 1-6 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 Serviceguard Hardware Configurations

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

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

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. This 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 must have the software support which enables and controls 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, Serviceguard switches to a standby LAN or moves affected packages to a standby 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 failure of other monitored resources, such as disk interfaces, a package can be moved to another node.
- 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 an Serviceguard cluster is 16. Fast/Wide SCSI disks or disk arrays can be connected to a maximum of 4 nodes at a time on a shared (multi-initiator) bus. Disk arrays using fibre channel and those that do not use a shared bus — such as the HP SureStore XP Series and the EMC Symmetrix — can be simultaneously connected to all 16 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 Fast/Wide SCSI disk or disk array can failover to a maximum of 4 nodes. A package that accesses data from a disk in a cluster using Fibre Channel or HP SureStore XP or EMC Symmetrix disk technology can be configured for failover among 16 nodes.

Note that 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 (regardless of disk technology). For instance, if a package only runs local executables, it can be configured to failover 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 the cables themselves are connected by a component such as a hub or a bridge. In the case of FDDI networks, each interface card is connected via a cable to a different concentrator. This arrangement of physical cables connected to each other via a bridge or concentrator or switch is known as a **bridged net**.

IP addresses can be associated with interfaces on a bridged net. An interface that has an IP address associated with it is known as a **primary interface**, and an interface that does not have an IP address associated with it is known as a **standby interface**. Standby interfaces are those which are available for switching by Serviceguard if a failure occurs on the primary. When Serviceguard detects a primary interface failure, it will switch the IP addresses and any associated connections from the failed interface card to a healthy standby interface card.

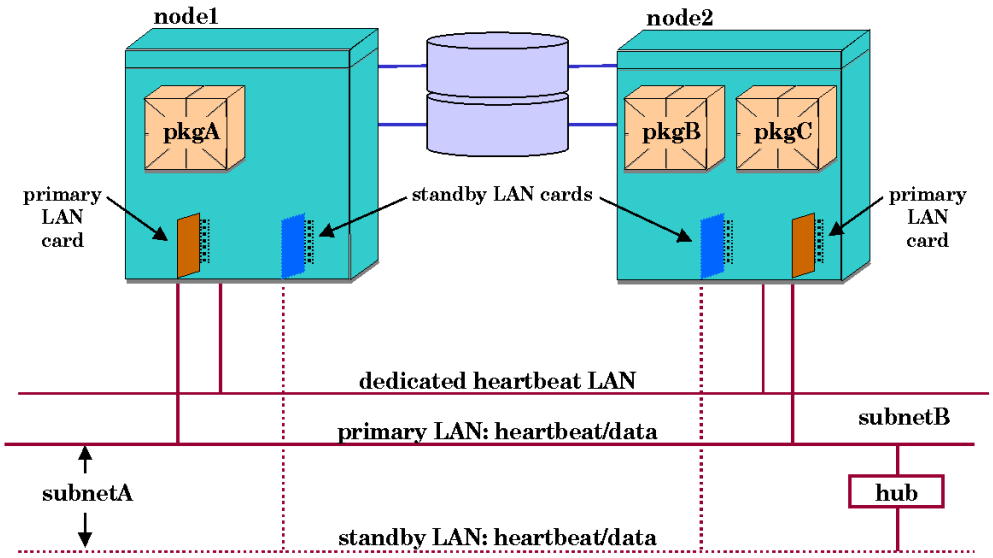
A selection of network configurations is described further in the following sections. For a complete list of supported networks, refer to the *Serviceguard Release Notes* for your version of the product.

**NOTE** Fibre Channel is no longer supported as a heartbeat or data LAN.

## Redundant Ethernet Configuration

The use of redundant network components is shown in Figure 2-1, which is an Ethernet configuration. Token ring is configured in a similar fashion.

**Figure 2-1 Redundant LANs**




In the figure, a two-node Serviceguard cluster has one bridged net configured with both a primary and a standby LAN card for the data/heartbeat subnet (Subnet A). Another LAN card provides an optional dedicated heartbeat LAN. Note that the primary and standby LAN segments are connected by a hub to provide a redundant data/heartbeat subnet. Each node has its own IP address for this subnet. In case of a failure of a primary LAN card for the data/heartbeat subnet, Serviceguard will perform a local switch to the standby LAN card on the same node.

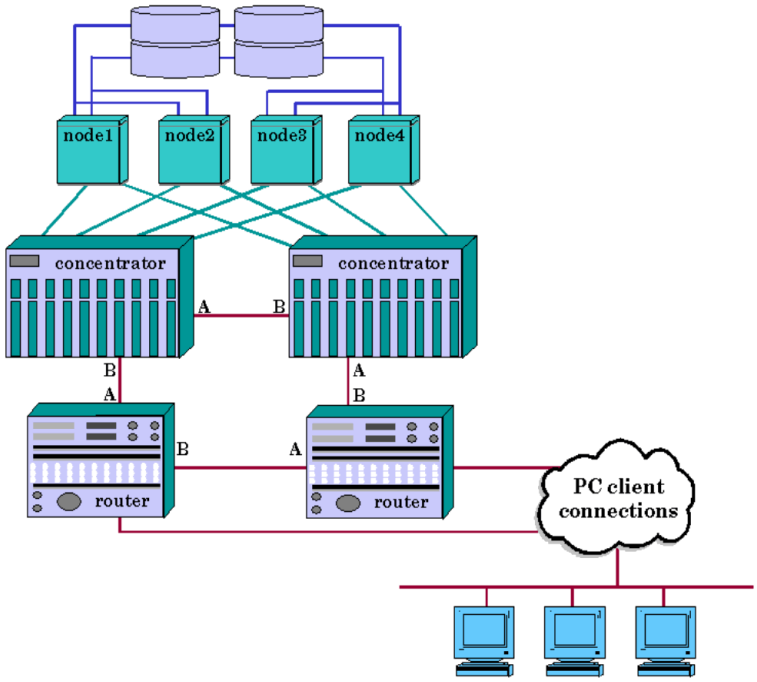
Redundant heartbeat is provided by the primary LAN and the dedicated LAN which are both carrying the heartbeat. In Figure 2-1, local switching is not needed for the dedicated heartbeat LAN, since there is already a redundant path via the other subnet. In case of data congestion on the primary LAN, the dedicated heartbeat LAN will prevent a false diagnosis of heartbeat failure. Each node has its own IP address for dedicated heartbeat LAN.

**NOTE** You should verify that network traffic is not too high on the heartbeat/data LAN. If traffic is too high, this LAN might not perform adequately in transmitting heartbeats if the dedicated heartbeat LAN fails.

## Providing Redundant FDDI Connections

FDDI is a high speed fiber optic interconnect medium. If you are using FDDI, you can create a redundant configuration by using a star topology to connect all the nodes to two concentrators, which are also connected to two routers, which communicate with the world outside the cluster. In this case, you use two FDDI cards in each node. The configuration is shown in Figure 2-2. Note that the concentrators are connected together using dual cables cross-connected Port A to Port B. The routers must be configured to send all packets to both concentrators.

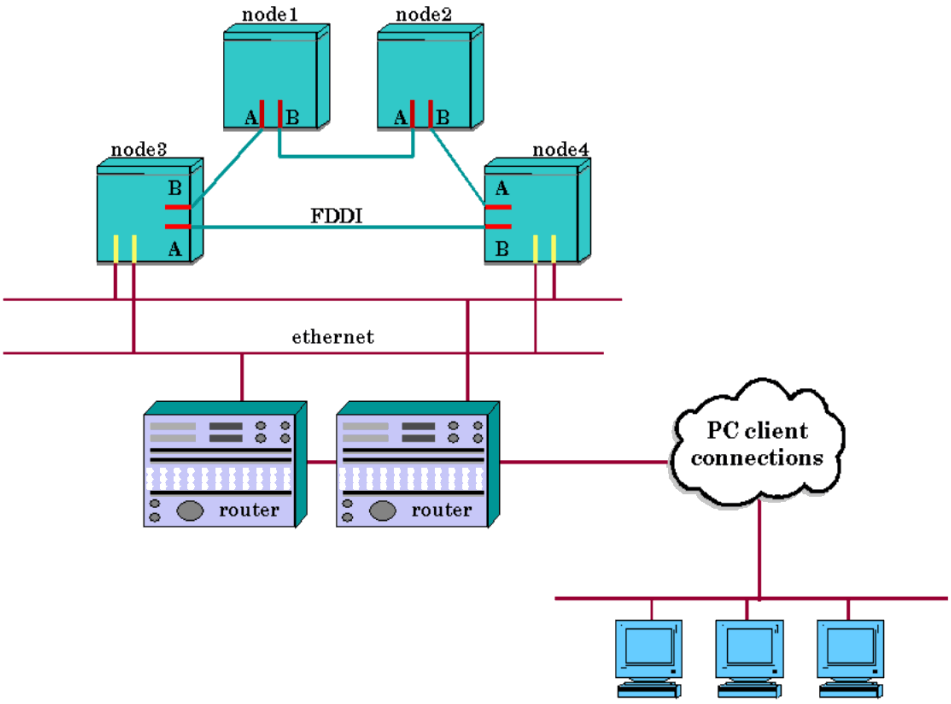
**Figure 2-2 Redundant FDDI Configuration**



## Using Dual Attach FDDI Stations

Another way of obtaining redundant FDDI connections is to configure dual attach stations on each node to create an FDDI ring, shown in Figure 2-3. An advantage of this configuration is that only one slot is used in the system card cage. In Figure 2-3, note that nodes 3 and 4 also use Ethernet to provide connectivity outside the cluster.

**Figure 2-3 Configuration with Dual Attach FDDI Stations**




The use of dual attach cards gives protection against failures in both cables and connectors, but does not protect against card failures. LAN card failure would result in a package switching to another node.

## Using a Serial (RS232) Heartbeat Line

Serviceguard supports a two-node configuration using serial (RS232) communication for heartbeats only. You select this as an alternate heartbeat interface to provide redundant heartbeat data.

**NOTE** The use of a serial (RS232) heartbeat line is supported only in a two-node cluster configuration. A serial heartbeat line is *required* in a two-node cluster that has only one heartbeat LAN. If you have at least two heartbeat LANs, or one heartbeat LAN and one standby LAN, a serial (RS232) heartbeat should *not* be used.

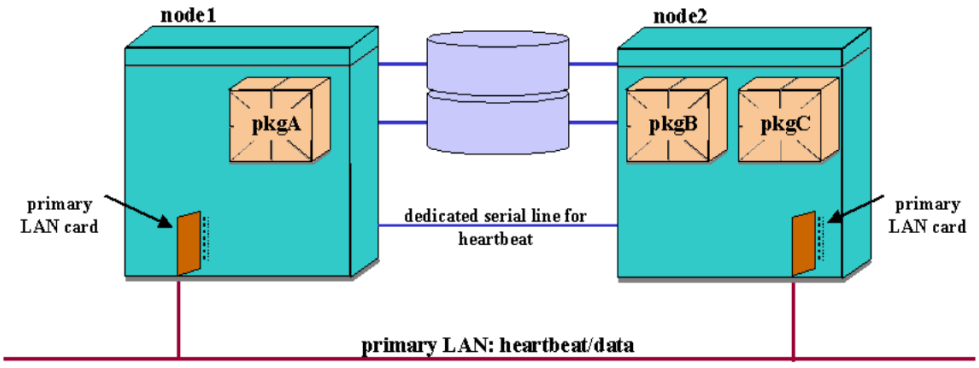
If the heartbeat network card fails on one node, having a serial line heartbeat keeps the cluster up just long enough to detect the LAN controller card status and to fail the node with bad network connections while the healthy node stays up and runs all the packages.

Even if you have a serial (RS232) line configured for redundant heartbeat, one LAN is still required to carry a heartbeat signal. The serial line heartbeat protects against network saturation but not against network failure, since Serviceguard requires TCP/IP to communicate between cluster members.

Serial (RS232) lines are inherently unreliable compared to network cards which run the TCP/IP protocol, such as Ethernet or FDDI. Unlike TCP/IP, the serial line protocol has no error correction or retry mechanism. Serial lines can also be complex and difficult to configure because of a lack of standards.

A serial (RS232) heartbeat line is shown in Figure 2-4.

**Figure 2-4 Serial (RS232) Heartbeat Line**



## Replacement of Failed Network Cards

Depending on the system configuration, it is possible to replace failed network cards while the cluster is running. The process is described under “Replacement of LAN Cards” in the chapter “Troubleshooting Your Cluster.”

# Redundant Disk Storage

Each node in a cluster has its own root disk, but each node is also 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 a Storage Manager, such as Logical Volume Manager (LVM), VERITAS Volume Manager (VxVM), or VERITAS Cluster Volume Manager (CVM). A disk storage group can be activated by no more than one node at a time, but when the package is moved, the storage group can be activated by the adoptive node. All of the disks in the storage group owned by a package must be connected to the original node and to all possible adoptive nodes for that package. Disk storage is made redundant by using RAID or software mirroring.

## Supported Disk Interfaces

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

- Single-ended SCSI.
- Fast/Wide SCSI.
- Fibre Channel.

Not all SCSI disks are supported. See the *HP Unix Servers Configuration Guide* (available through your HP representative) for a list of currently supported disks.

**NOTE** In a cluster that contains systems with PCI SCSI adapters, you cannot attach both PCI and NIO SCSI adapters to the same shared SCSI bus.

External shared Fast/Wide SCSI buses must be equipped with in-line terminators for disks on a shared bus. Refer to the “Troubleshooting” chapter for additional information.

When planning and assigning SCSI bus priority, remember that one node can dominate a bus shared by multiple nodes, depending on what SCSI addresses are assigned to the controller for each node on the

shared bus. All SCSI addresses, including the addresses of all interface cards, must be unique for all devices on a shared bus. See the manual *Configuring HP-UX for Peripherals* for information on SCSI bus addressing and priority.

## Data Protection

It is required that you provide data protection for your highly available system, using one of two methods:

- Disk Mirroring
- Disk Arrays using RAID Levels and Multiple Data Paths

### Disk Mirroring

Disk mirroring is one method for providing data protection. The logical volumes used for Serviceguard packages should be mirrored. Serviceguard does not provide protection for data on your disks. This capability is provided for LVM storage with HP's MirrorDisk/UX product, and for VxVM and CVM with the VERITAS Volume Manager (B9116AA). When you configure logical volumes using software mirroring, the members of each mirrored set contain exactly the same data. If one disk should fail, the storage manager will automatically keep the data available by accessing the other mirror. Three-way mirroring in LVM (or additional plexes with VxVM) may be used to allow for online backups or even to provide an additional level of high availability.

To protect against Fibre Channel or SCSI bus failures, each copy of the data must be accessed by a separate bus; that is, you cannot have all copies of the data on disk drives connected to the same bus.

It is critical for high availability that you mirror both data and root disks. If you do not mirror your data disks and there is a disk failure, you will not be able to run your applications on any node in the cluster until the disk has been replaced and the data reloaded. If the root disk fails, you will be able to run your applications on other nodes in the cluster, since the data is shared. However, system behavior at the time of a root disk failure is unpredictable, and it is possible for an application to hang while the system is still running, preventing it from being started on another node until the failing node is halted. Mirroring the root disk can allow the system to continue normal operation when a root disk failure occurs, and help avoid this downtime.

### Disk Arrays using RAID Levels and Multiple Data Paths

An alternate method of achieving protection for your data is to employ a disk array with hardware RAID levels that provide data redundancy, such as RAID Level 1 or RAID Level 5. The array provides data redundancy for the disks. This protection needs to be combined with the use of redundant host bus interfaces (SCSI or Fibre Channel) between each node and the array. The use of redundant interfaces, configured with LVM's PV Links feature or VxVM's dynamic multipathing (DMP), protects against single points of failure in the I/O channel, and RAID 1 or 5 configuration provides redundancy for the storage media. (PV links are also known as alternate links in LVM, or multiple paths in VxVM.)

DMP is available as a separate component of VxVM. DMP for active/active devices requires B9116AA, but DMP for active/passive devices is available for no charge with the base product, Base-VxVM.

## Monitoring of Disks Through Event Monitoring Service

If you are using LVM, you can configure disk monitoring to detect a failed mechanism by using the disk monitor capabilities of the EMS HA Monitors, available as a separate product (B5736DA). Monitoring can be set up to trigger a package failover or to report disk failure events to a Serviceguard, to another application, or by email. For more information, refer to the manual *Using High Availability Monitors* (B5736-90042), available at http://docs.hp.com -> High Availability.

## Replacement of Failed Disk Mechanisms

Mirroring provides data protection, but after a disk failure, the failed disk must be replaced. With conventional disks, this is done by bringing down the cluster and replacing the mechanism. With disk arrays and with special HA disk enclosures, it is possible to replace a disk while the cluster stays up and the application remains online. The process is described under "Replacing Disks" in the chapter "Troubleshooting Your Cluster."

## Replacement of Failed I/O Cards

Depending on the system configuration, it is possible to replace failed disk I/O cards while the system remains online. The process is described under “Replacing I/O Cards” in the chapter “Troubleshooting Your Cluster.”

## Sample SCSI Disk Configurations

Figure 2-5 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. Please note that Package A’s disk and the mirror of Package B’s disk are on one interface while Package B’s disk and the mirror of Package A’s disk are on a separate bus. This arrangement eliminates single points of failure and makes either the disk or its mirror available in the event one of the buses fails.

**Figure 2-5 Mirrored Disks Connected for High Availability**

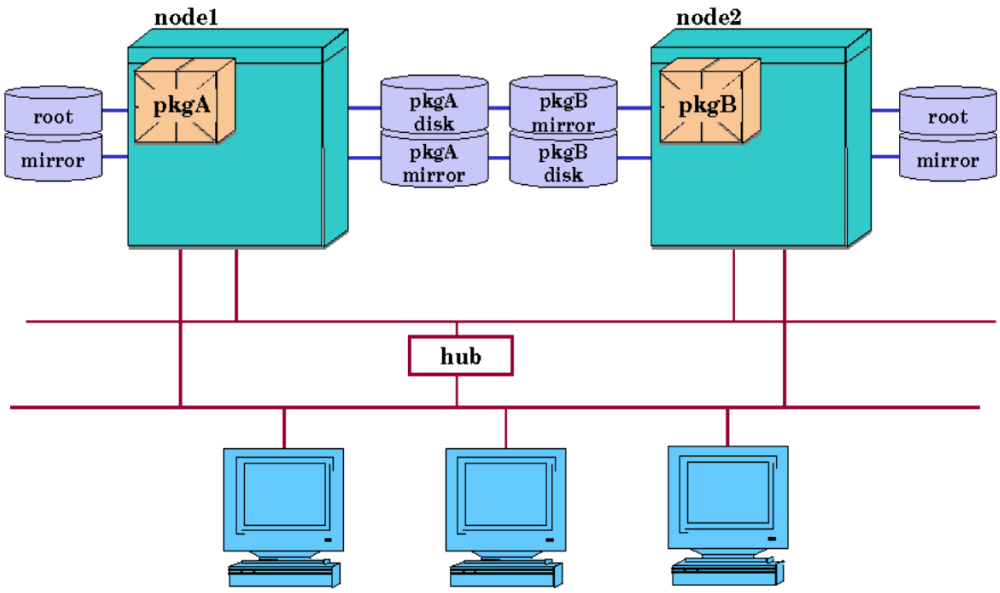
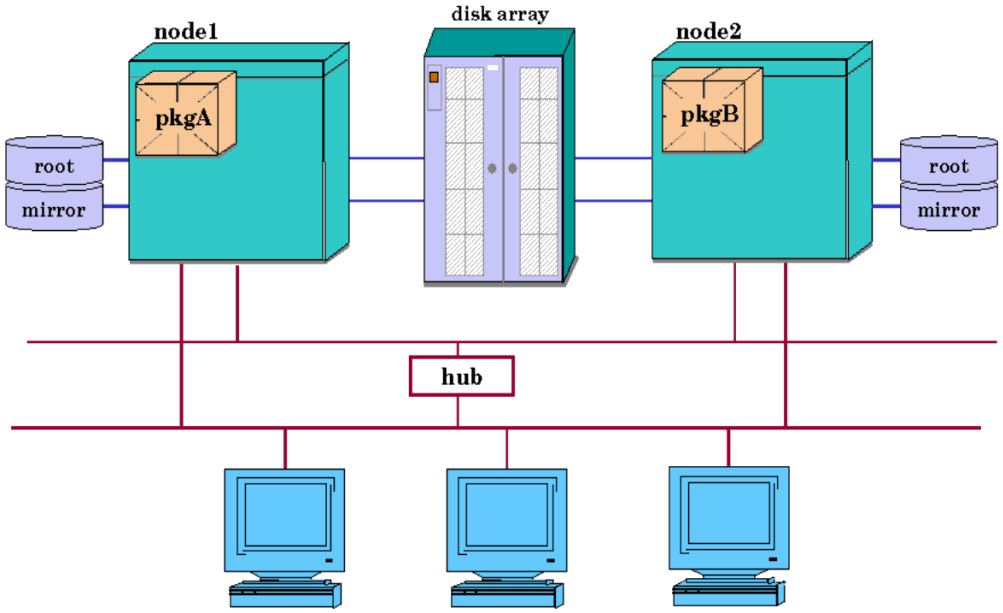



Figure 2-6 below shows a similar cluster with a disk array connected to each node on two I/O channels. This kind of configuration can use LVM's PV Links or other multipath software such as VERITAS Dynamic Multipath (DMP) or EMC PowerPath

**Figure 2-6 Cluster with High Availability Disk Array**

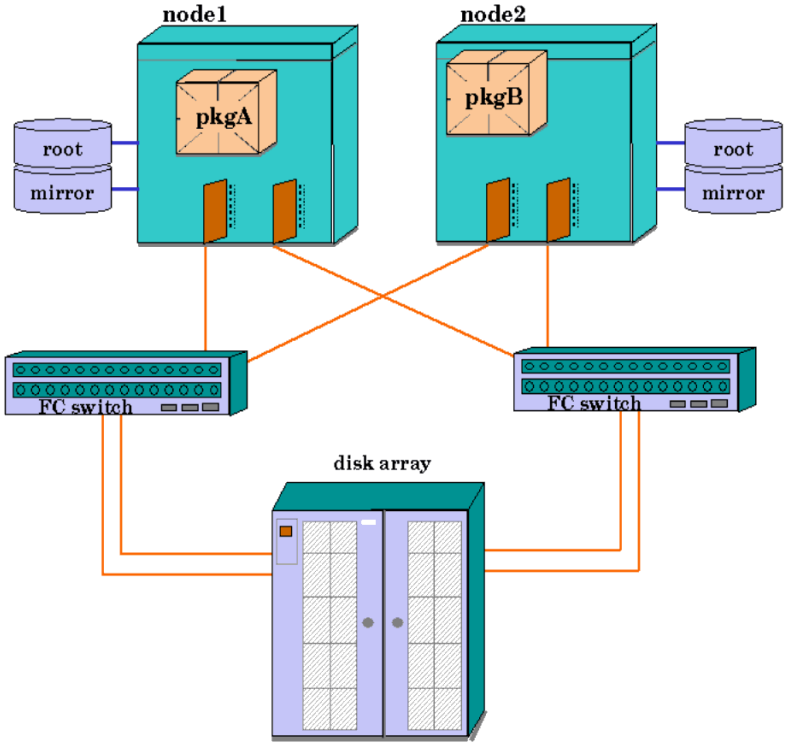



Details on logical volume configuration for Serviceguard, including PV Links, are given in the chapter "Building an HA Cluster Configuration."

## Sample Fibre Channel Disk Configuration

In Figure 2-7 below, the root disks are shown with simple mirroring, but the shared storage is now accessed via redundant Fibre Channel switches attached to a disk array. The cabling is set up so that each node is attached to both switches, and both switches are attached to the disk array with redundant links.

**Figure 2-7** **Cluster with Fibre Channel Switched Disk Array**



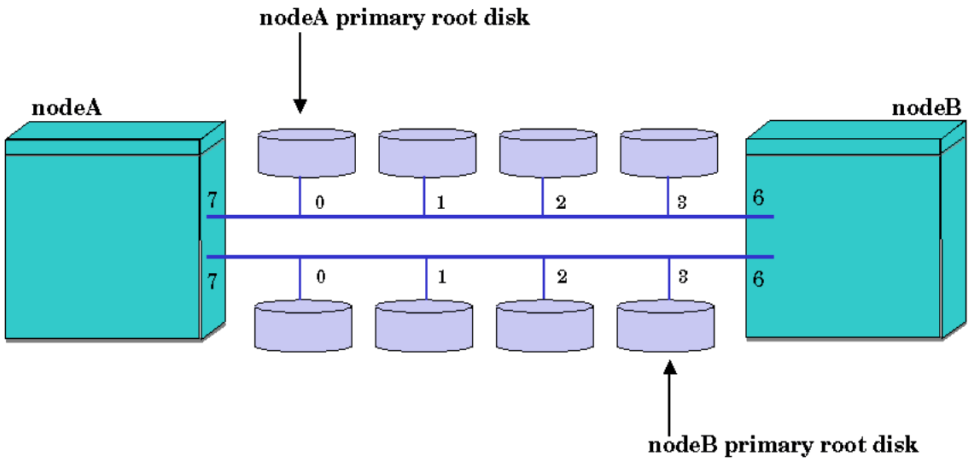

This type of configuration also uses PV links or other multipath software such as VERITAS Dynamic Multipath (DMP) or EMC PowerPath.

## Root Disk Limitations on Shared SCSI Buses

The IODC firmware does not support two or more nodes booting from the same SCSI bus at the same time. For this reason, it is important not to attach more than one root disk per cluster to a single SCSI bus.

For example, Figure 2-8 shows a supported configuration in which two nodes share an external SCSI bus and Node A has its primary root disk connected to the bus, but node B has its primary root disk connected to a different bus. (Numbers 0 to 3, 6 and 7 are SCSI addresses on the different buses.)

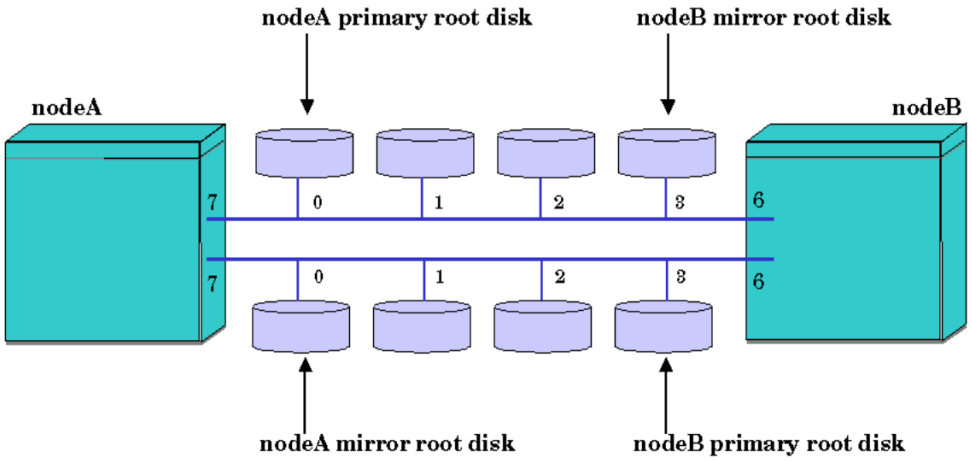
**Figure 2-8 Root Disks on Different Shared Buses**




Note that if both nodes had their primary root disks connected to the *same* bus, you would have an unsupported configuration.

You *can* put a mirror copy of Node B's root disk on the same SCSI bus as Node A's primary root disk, because three failures would have to occur for both systems to boot at the same time, which is an acceptable risk. In such a scenario, Node B would have to lose its primary root disk and be rebooted, and Node A would have to be rebooted at the same time Node B is, for the IODC firmware to run into a problem. This configuration is shown in Figure 2-9.

**Figure 2-9 Primaries and Mirrors on Different Shared Buses**




Note that you *cannot* use a disk within a disk array as a root disk if the array is on a shared bus.

## 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 mirror copies are attached to different power sources. The boot disk should be powered from the same circuit as its corresponding node.

In particular, the cluster lock disk (used as a tie-breaker when re-forming a cluster) should have a redundant power supply, or else it can be powered from a different supply than the nodes in the cluster. Your HP representative can provide more details about the layout of power supplies, disks, and LAN hardware for clusters.

Many current disk arrays and other racked systems contain multiple power inputs, which should be deployed so that the different power inputs on the device are connected to separate power circuits.Devices with two or three power inputs generally can continue to operate normally if no more than one of the power circuits has failed. Therefore, if all of the hardware in the cluster has 2 or 3 power inputs, then at least three separate power circuits will be required to ensure that there is no single point of failure in the power circuit design for the cluster.

# Larger Clusters

You can create clusters of up to 16 nodes with Serviceguard. Clusters of up to 16 nodes may be built by connecting individual SPUs via Ethernet; and you can configure up to 8 systems as an Serviceguard cluster using FDDI networking.

The possibility of configuring a cluster consisting of 16 nodes does not mean that all types of cluster configuration behave in the same way in a 16-node configuration. For example, in the case of shared F/W SCSI buses, the practical limit on the number of nodes that can be attached to the same shared bus is four, because of bus loading and limits on cable length. Even in this case, 16 nodes could be set up as an administrative unit, and sub-groupings of four could be set up on different SCSI buses which are attached to different mass storage devices.

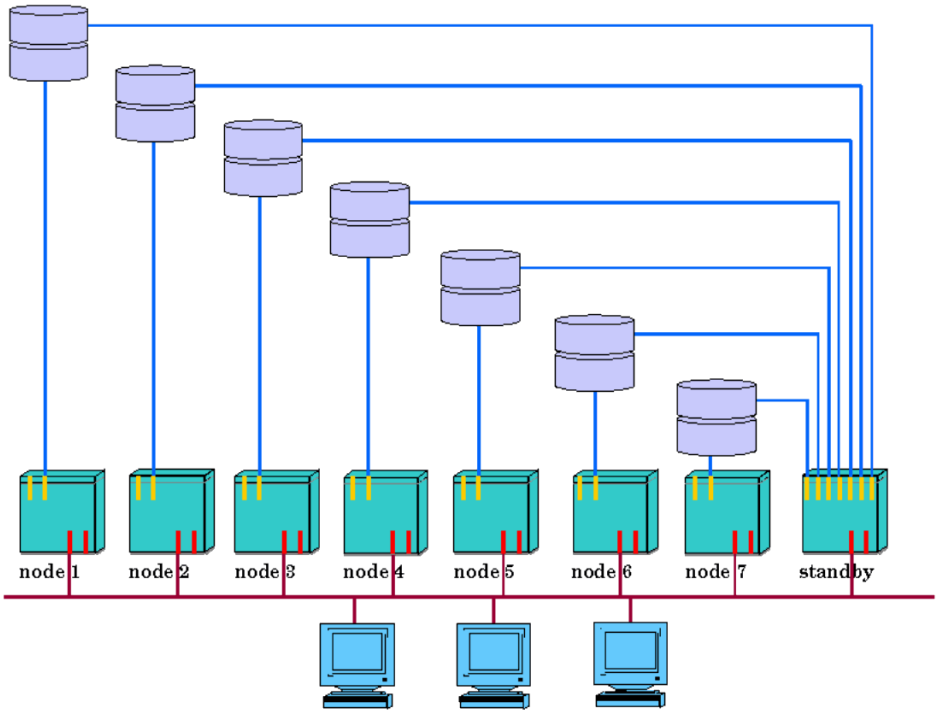
In the case of non-shared F/W SCSI connections to an XP series or EMC disk array, the four-node limit does not apply. Each node can be connected directly to the XP or EMC by means of two F/W SCSI buses supporting PV links. Packages can be configured to fail over among all sixteen nodes. For more about this type of configuration, see “Point to Point Connections to Storage Devices,” below.

**NOTE** When configuring larger clusters, be aware that cluster and package configuration times as well as execution times for commands such as `cmviewcl` will be extended. Some command options help to remedy this situation. For more information, refer to the man page for `cmquerycl`.

## Active/Standby Model

You can also create clusters in which there is a standby node. For example, an eight node configuration in which one node acts as the standby for the other seven could easily be set up by equipping the backup node with seven shared buses allowing separate connections to each of the active nodes. This configuration is shown in Figure 2-10.

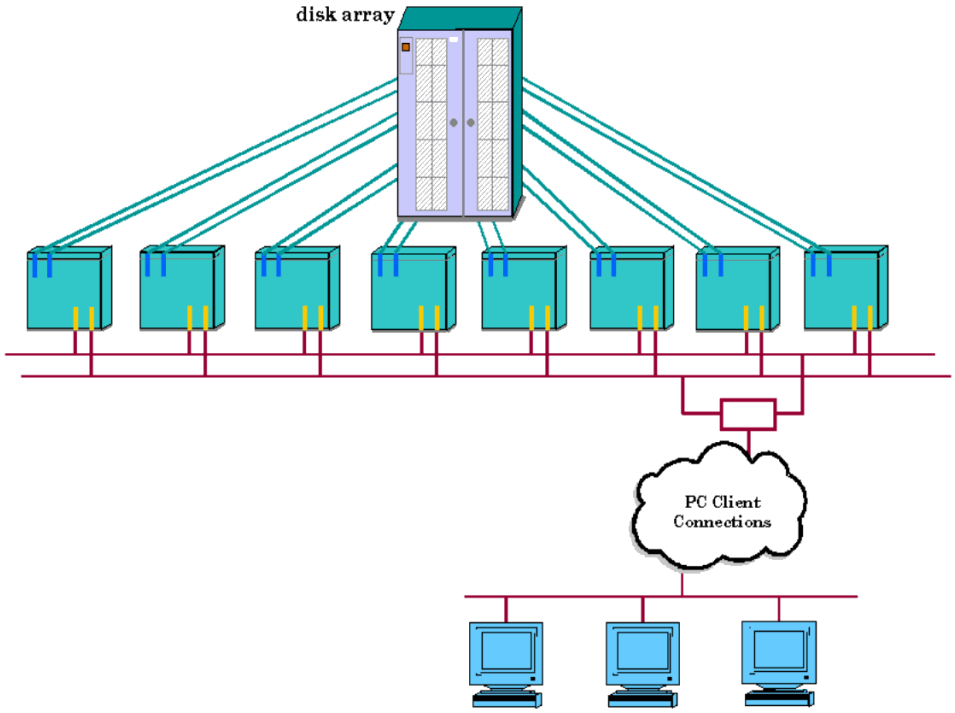
**Figure 2-10 Eight-Node Active/Standby Cluster**




## Point to Point Connections to Storage Devices

Some storage devices allow point-to-point connection to a large number of host nodes without using a shared SCSI bus. An example is shown in Figure 2-11, a cluster consisting of eight nodes with a Fibre Channel interconnect. (Client connection is provided through Ethernet.) The nodes access shared data on an XP 256 or EMC disk array configured with 16 I/O ports. Each node is connected to the array using two separate F/W SCSI channels configured with PV Links. Each channel is a dedicated bus; there is no daisy-chaining.

**Figure 2-11 Eight-Node Cluster with XP or EMC Disk Array**




Fibre Channel switched configurations also are supported using either an arbitrated loop or fabric login topology. For additional information about supported cluster configurations, refer to the *HP Unix Servers Configuration Guide*, available through your HP representative.

# 3 Understanding Serviceguard Software Components

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

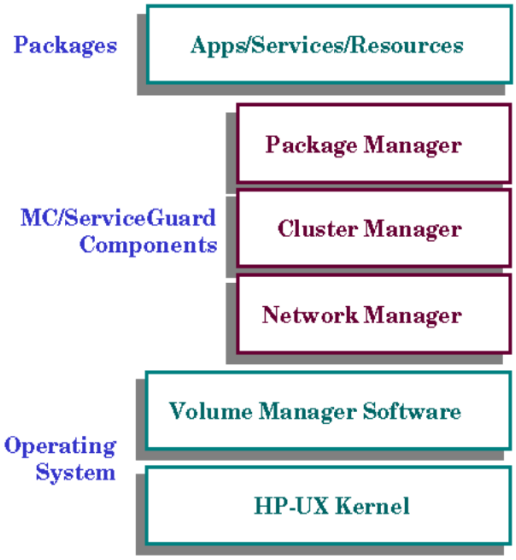
- Serviceguard Architecture
- How the Cluster Manager Works
- How the Package Manager Works
- How Package Control Scripts Work
- How the Network Manager Works
- Volume Managers for Data Storage
- Responses to Failures

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. This chapter discusses these components in some detail.

**Figure 3-1 Serviceguard Software Components**




## Serviceguard Daemons

There are nine daemon processes associated with Serviceguard. They are:

- `/usr/lbin/cmclconfd`—Serviceguard Configuration Daemon
- `/usr/lbin/cmcld`—Serviceguard Cluster Daemon
- `/usr/lbin/cmlogd`—Serviceguard Syslog Log Daemon
- `/usr/lbin/cmlvmd`—Cluster Logical Volume Manager Daemon
- `/opt/cmom/lbin/cmomd`—Cluster Object Manager Daemon
- `/usr/lbin/cmsnmpd`—Cluster SNMP subagent (optionally running)

- /usr/lbin/cmsrvassistd—Serviceguard Service Assistant Daemon
- /usr/lbin/cmtaped—Serviceguard Shared Tape Daemon
- /usr/lbin/qs—Serviceguard Quorum Server Daemon

Each of these daemons logs to the /var/adm/syslog/syslog.log file except for the quorum server, which logs to the standard output (it is suggested you redirect output to a file named /var/adm/qs/qs.log) and /opt/cmom/lbin/cmomd, which logs to /var/opt/cmom/cmomd.log.

### 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 inetd(1M). There are entries in the /etc/inetd.conf file.

### Cluster Daemon: cmcld

This daemon is used to determine cluster membership by sending heartbeat messages to other cmcld daemons on other nodes within the Serviceguard cluster. It runs at a real time priority and is locked in memory. The cmcld daemon sets a **safety timer** in the kernel which is used to detect kernel hangs. If this timer is not reset periodically by cmcld, the kernel will cause a system TOC, that is, a Transfer of Control, which means a CPU reset and the creation of a crash dump file. This could occur because cmcld could not communicate with the majority of the cluster's members, or because cmcld 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 /var/adm/syslog/syslog.log and the kernel's message buffer.

The cmcld daemon also detects the health of the networks on the system and performs local lan failover. Finally, this daemon handles the management of Serviceguard packages, determining where to run them and when to start them.

**NOTE** The three central components of Serviceguard—Package Manager, Cluster Manager, and Network Manager—run as parts of the cmcld daemon. This daemon runs at priority 20 on all cluster nodes. It is

important that user processes should have a lower priority than 20, otherwise they might prevent Serviceguard from updating the kernel safety timer, thus causing the node to undergo a TOC.

### Syslog Log Daemon: cmlogd

`cmlogd` is used by `cmcld` to write messages to syslog. Any message written to syslog by `cmcld` it written through `cmlogd`. This is to prevent any delays in writing to syslog from impacting the timing of `cmcld`.

### Cluster Logical Volume Manager Daemon: cmlvmd

This daemon is responsible for keeping track of all the volume group(s) that have been made cluster aware. When a volume group is made cluster aware, a cluster node can only activate it in exclusive mode. This prevents the volume group from being activated in write mode by more than one node at a time.

### 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 the object manager 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 `inetd(1M)`. There are entries in the `/etc/inetd.conf` file.

### Cluster SNMP Agent Daemon: cmsnmpd

This daemon is used by the Serviceguard graphic interface (Serviceguard Manager) to obtain information about the cluster. This daemon produces the Cluster MIB. Details on the cluster MIB can be found on URL http://docs.hp.com/hpux/ha under the Frequently Asked Questions area.

This will only be running if the `/etc/rc.config.d/cmsnmpagt` file has been edited to autostart this subagent. For proper execution, the cmsnmpd has to start before the Serviceguard cluster comes up.

### Service Assistant Daemon: cmsrvassistd

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

- Executing package run and halt scripts
- Launching services

For services, `cmcld` monitors the service process and, depending on the number of service retries, `cmcld` either restarts the service through `cmsrvassistd` or it causes the package to halt and moves the package to an available alternate node.

### Shared Tape Daemon: cmtaped

The shared tape daemon is responsible for keeping track of all the shared tape devices that are part of the cluster. Share tape devices can be configured using the `stapplyconf` command.

### 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.

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.

# 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 using Serviceguard Manager or 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 monitored TCP/IP network or RS232 serial line configured as a heartbeat device. (LAN monitoring is further discussed later in the section "Monitoring LAN Interfaces and Detecting Failure.")

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 or running heartbeat over a serial (RS232) line. 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.

**NOTE** You cannot run heartbeat on a serial line by itself. Refer to Chapter 2, "Using a Serial (RS232) Heartbeat Line," for details about serial lines in Serviceguard.

**IMPORTANT** Multiple heartbeats are sent in parallel. It is normally recommended that you configure all subnets that interconnect cluster nodes as heartbeat networks, since this increases protection against multiple faults at no additional cost. However, if you will be using the VERITAS Cluster Volume Manager (CVM), you can use only a single heartbeat subnet. In this case, the heartbeat should be configured with standby LANs or as a group of aggregated ports. See below, "Single Heartbeat Subnet Required with CVM."

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 in Serviceguard Manager or 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 *`cmcld`* 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 `/etc/rc.config.d/cmcluster` 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 disk** or a **quorum server**.

The cluster lock is used as a tie-breaker only for situations in which a running cluster fails and, as Serviceguard attempts to form a new cluster, the cluster is split into two sub-clusters of equal size. Each sub-cluster will attempt to acquire the cluster lock. The sub-cluster which gets the cluster lock will form the new cluster, preventing the

possibility of two sub-clusters running at the same time. If the two sub-clusters are of unequal size, the sub-cluster with greater than 50% of the nodes will form the new cluster, and the cluster lock is not used.

If you have a two-node cluster, you are required to configure a cluster lock. 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 halt or perform a TOC. Without a cluster lock, a failure of either node in the cluster will cause the other node, and therefore the cluster, to halt. Note also that if the cluster lock fails during an attempt to acquire it, the cluster will halt.

### Lock Requirements

A one-node cluster does not require a cluster lock. A two-node cluster *requires* a cluster lock. In clusters larger than 3 nodes, a cluster lock is strongly recommended. If you have a cluster with more than four nodes, a cluster lock disk is not allowed, but a quorum server may be used.

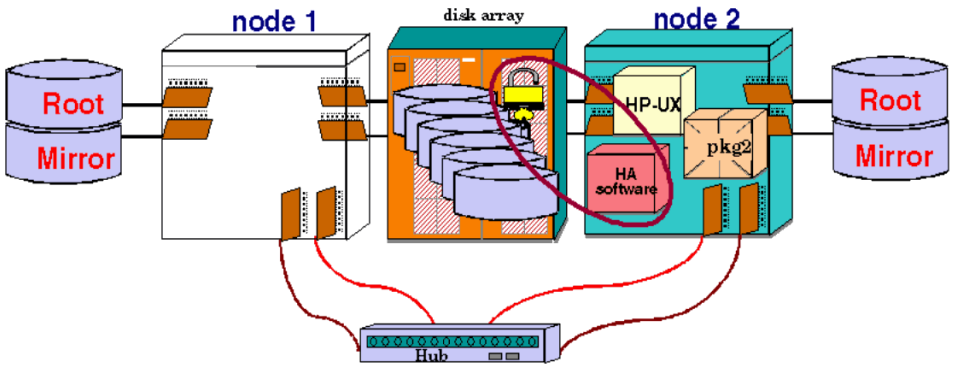
## Use of an LVM Lock Disk as the Cluster Lock

A lock disk may be used for clusters up to and including four nodes in size. The cluster lock disk is a special area on an LVM disk located in a volume group 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 disk is not dedicated for use as the cluster lock; the disk can be employed as part of a normal volume group with user data on it. The cluster lock volume group and physical volume names are identified in the cluster configuration file.

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

**Figure 3-2 Lock Disk Operation**




Serviceguard periodically checks the health of the lock disk 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 choose between two lock disk options—a single or dual lock disk—based on the kind of high availability configuration you are building. *A single lock disk is recommended where possible*. With both single and dual locks, however, it is important that the cluster lock be available even if the power circuit to one node fails; thus, the choice of a lock configuration depends partly on the number of power circuits available. Regardless of your choice, all nodes in the cluster must have access to the cluster lock to maintain high availability.

### Single Lock Disk

It is recommended that you use a single lock disk. A single lock disk should be configured on a power circuit separate from that of any node in the cluster. For example, it is highly recommended to use three power circuits for a two-node cluster, with a single, separately powered disk for the cluster lock. For two-node clusters, this single lock disk may not share a power circuit with either node, and it must be an external disk. For three or four node clusters, the disk should not share a power circuit with 50% or more of the nodes.

### Dual Lock Disk

If you are using disks that are internally mounted in the same cabinet as the cluster nodes, then a single lock disk would be a single point of failure in this type of cluster, since the loss of power to the node that has the lock disk in its cabinet would also render the cluster lock unavailable. Similarly, in a campus cluster, where the cluster contains nodes running in two separate data centers, a single lock disk would be a single point of failure should the data center it resides in suffer a catastrophic failure. *In these two cases only*, a dual cluster lock, with two separately powered cluster disks, should be used to eliminate the lock disk as a single point of failure. For a dual cluster lock, the disks must not share either a power circuit or a node chassis with one another. In this case, if there is a power failure affecting one node and disk, the other node and disk remain available, so cluster re-formation can take place on the remaining node. For a campus cluster, there should be one lock disk in each of the data centers, and all nodes must have access to both lock disks. In the event of a failure of one of the data centers, the nodes in the remaining data center will be able to acquire their local lock disk, allowing them to successfully reform a new cluster.

**NOTE**

*A dual lock disk does not provide a redundant cluster lock.* In fact, the dual lock is a *compound* lock. This means that *two* disks must be available at cluster formation time rather than the one that is needed for a single lock disk. Thus, the *only recommended usage* of the dual cluster lock is when the single cluster lock cannot be isolated at the time of a failure from exactly one half of the cluster nodes.

If one of the dual lock disks fails, Serviceguard will detect this when it carries out periodic checking, and it will write a message to the syslog file. After the loss of one of the lock disks, the failure of a cluster node could cause the cluster to go down.

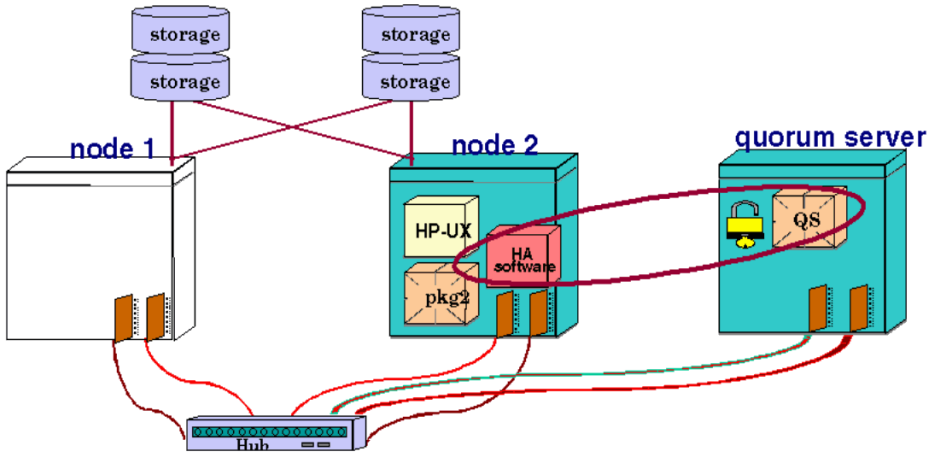
## Use of the Quorum Server as the Cluster Lock

A quorum server can be used in clusters of any size. 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." If communications are lost between two equal-sized groups of nodes, the group that obtains the lock from the Quorum Server will take over the cluster and the other nodes will perform a TOC. Without a cluster lock, a failure of either group of nodes will cause the other group, and therefore the cluster, to halt. Note also that 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**




The quorum server runs on a separate system, and can provide quorum services for multiple clusters.

## No Cluster Lock

Normally, you should not configure a cluster of three or fewer nodes without a cluster lock. In two-node clusters, a cluster lock is required. You may consider using no cluster lock with configurations of three or more nodes, although the decision should be affected by the fact that any cluster may require tie-breaking. For example, if one node in a three-node cluster is removed for maintenance, the cluster reforms as a two-node cluster. If a tie-breaking scenario later occurs due to a node or communication failure, the entire cluster will become unavailable.

In a cluster with four or more nodes, you may not need a cluster lock since the chance of the cluster being split into two halves of equal size is very small. However, be sure to configure your cluster to prevent the failure of exactly half the nodes at one time. For example, make sure there is no potential single point of failure such as a single LAN between equal numbers of nodes, or that you don't have exactly half of the nodes on a single power circuit.

# 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.

## Package Types

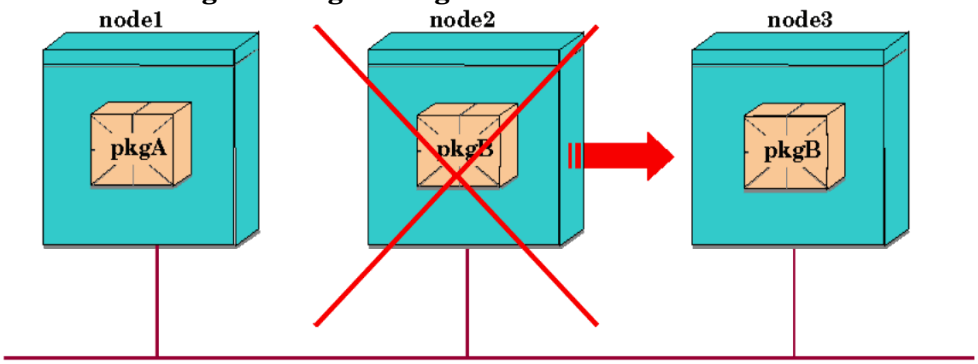
Two different types of packages can run in the cluster: the **failover package** and the **system multi-node package**. The system multi-node package is used only on systems that employ VERITAS Cluster Volume Manager (CVM) as a storage manager. This package, known as `VxVM-CVM-pkg`, runs on all nodes that are active in the cluster and provides cluster membership information to the volume manager software. This type of package is configured and used only when you employ CVM for storage management. The process of creating the system multi-node package for CVM is described in Chapter 5. The rest of this section describes the standard failover packages.

## Failover Packages

A failover 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 3-4 Package Moving During Failover**




## Configuring Packages

Each package is separately configured. You create a package by using Serviceguard Manager or 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 subnet 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 subnets, all of them must be available on the target node before the package will be started.

The switching of relocatable IP addresses is shown in Figure 3-5 and Figure 3-6. Figure 3-5 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 3-5 Before Package Switching**

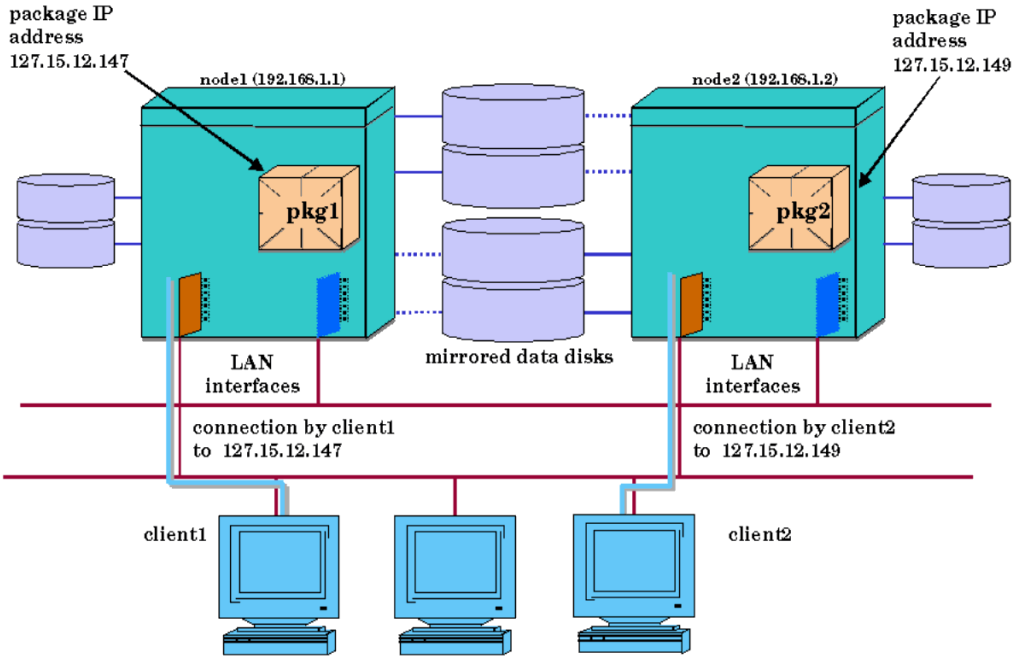
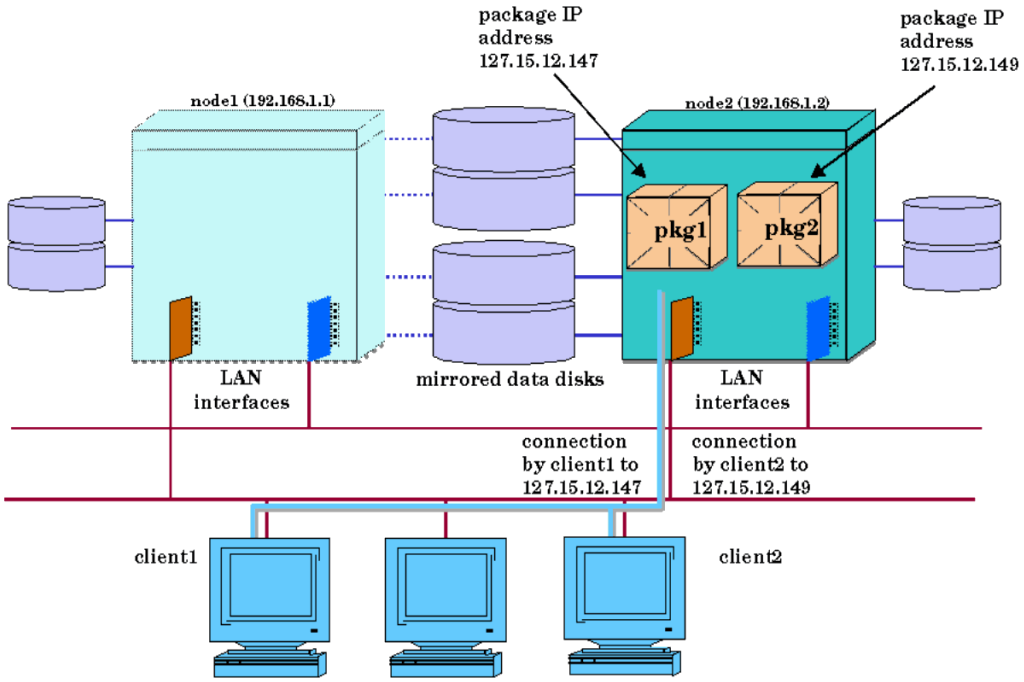



Figure 3-6 shows the condition where Node 1 has failed and Package 1 has been transferred to Node 2. Package 1's IP address was transferred to Node 2 along with the package. Package 1 continues to be available and is now running on Node 2. Also note that Node 2 can now access both Package A's disk and Package B's disk.

**Figure 3-6 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 or set with Serviceguard Manager. 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

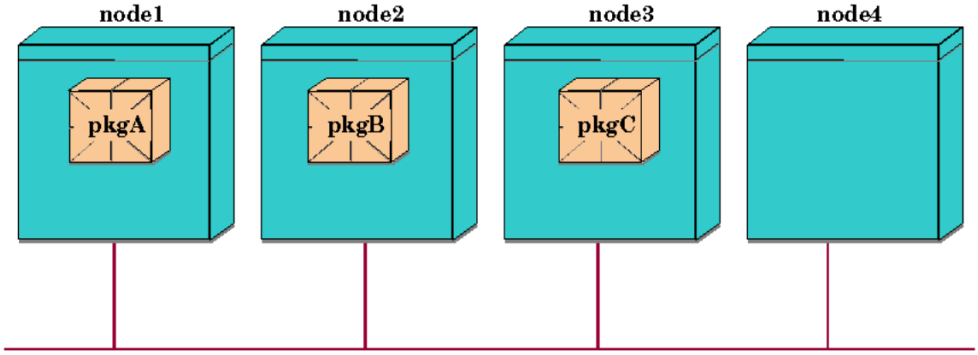
Using the MIN_PACKAGE_NODE failover policy, it is possible to configure a cluster that lets you use one node as an **automatic rotating standby** node for the cluster. Consider the following package configuration for a four node cluster. Note that all packages can run on all nodes and have the same NODE_NAME 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 |

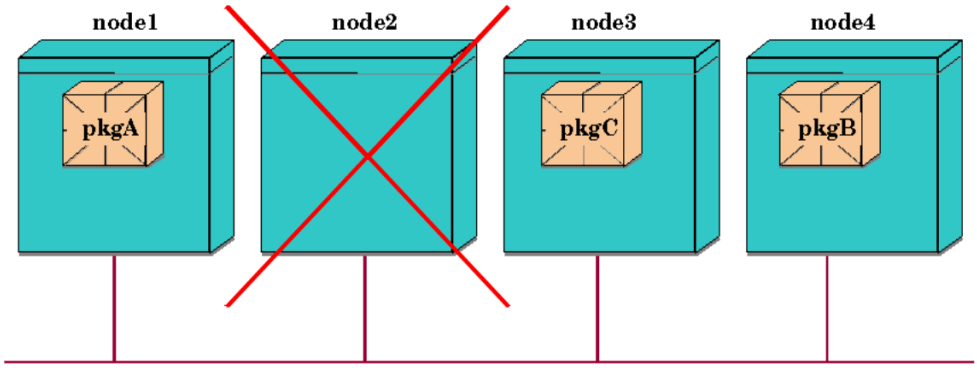
When the cluster starts, each package starts as shown in Figure 3-7.

**Figure 3-7 Rotating Standby Configuration before Failover**




If a failure occurs, any package would fail over to the node containing fewest running packages, as in Figure 3-8, which shows a failure on node 2:

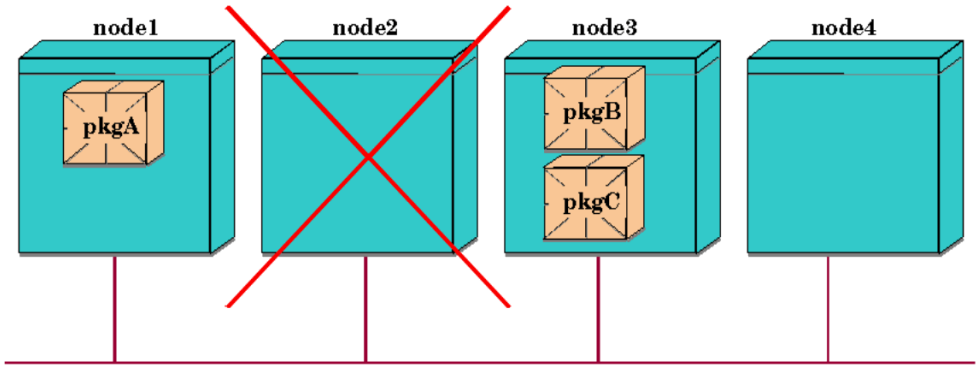
**Figure 3-8 Rotating Standby Configuration after Failover**




**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 3-7, but the failure of node 2 would cause the package to start on node 3, as in Figure 3-9:

**Figure 3-9** `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 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 it 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 3-10 Automatic Failback Configuration before Failover**

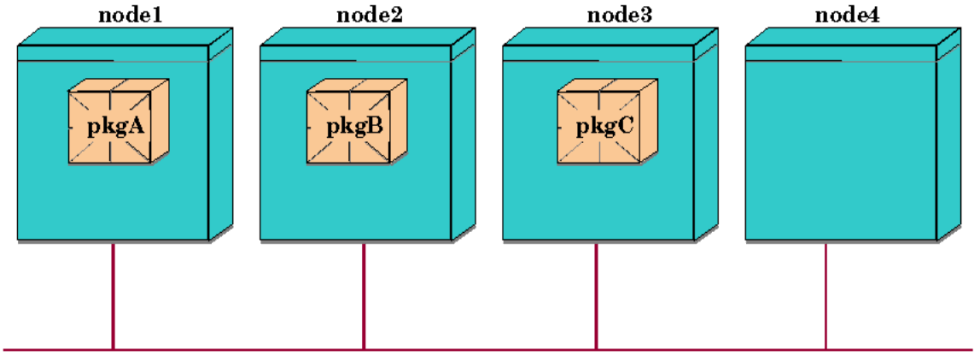
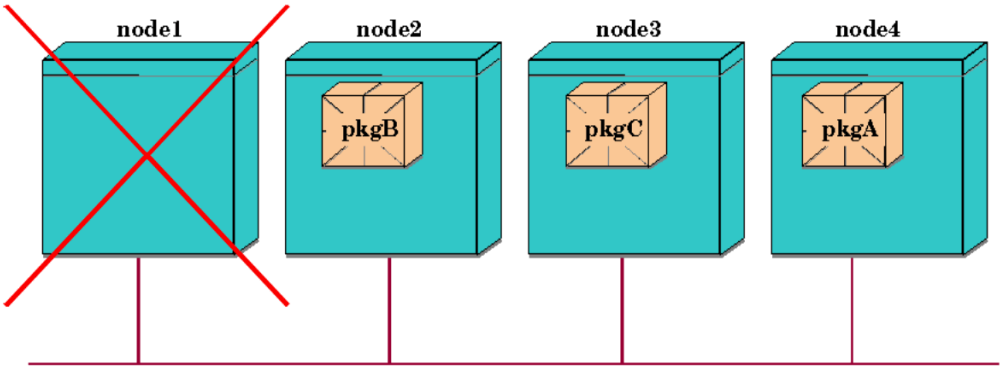



**Table 3-2 Node Lists in Sample Cluster**

| 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 |

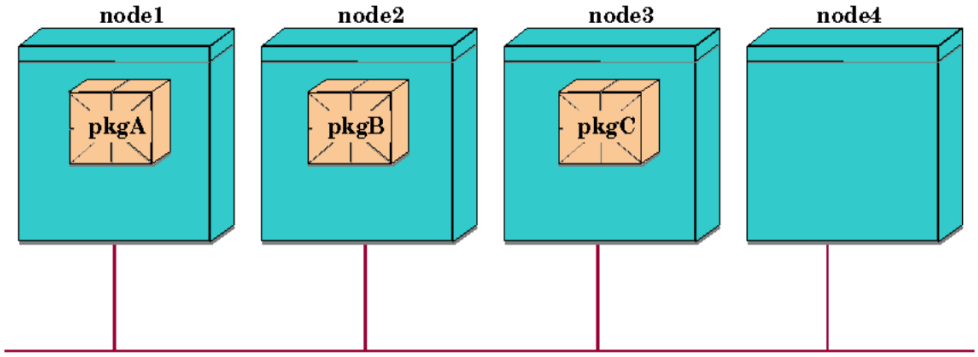
Node1 panics, and after the cluster reforms, pkgA starts running on node 4:

**Figure 3-11 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 3-12 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.

## Using Older Package Configuration Files

If you are using package configuration files that were generated using a previous version of Serviceguard, then the FAILOVER_POLICY will be the default package behavior of CONFIGURED_NODE and the FAILBACK_POLICY will be the default package behavior of MANUAL. If you wish to change these policies, edit the package configuration file to add the parameters, or use cmmakepkg to create a new package configuration file.

Starting with the A.11.12 version of Serviceguard, the PKG_SWITCHING_ENABLED parameter was renamed AUTO_RUN, and the NET_SWITCHING_ENABLED parameter was renamed to LOCAL_LAN_FAILOVER_ALLOWED. The older names will still work in your configuration files, but it is recommended to change the keywords.

## Using the Event Monitoring Service

Basic package resources include cluster nodes, LAN interfaces, and services, which are the individual processes within an application. All of these are monitored by ServiceGuard directly. In addition, you can use the Event Monitoring Service registry through which add-on monitors can be configured. This registry allows other software components to supply monitoring of their resources for ServiceGuard. Monitors currently supplied with other software products include EMS High Availability Monitors, and an ATM monitor.

If a registered resource is configured in a package, the package manager calls the resource registrar to launch an external monitor for the resource. Resources can be configured to start up either at the time the node enters the cluster or at the end of package startup. The monitor then sends messages back to ServiceGuard, which checks to see whether the resource is available before starting the package. In addition, the package manager can fail the package to another node or take other action if the resource becomes unavailable after the package starts.

You can specify a registered resource for a package by selecting it from the list of available resources displayed in the Serviceguard Manager Configuring Packages. The size of the list displayed by Serviceguard Manager depends on which resource monitors have been registered on your system. Alternatively, you can obtain information about registered resources on your system by using the command `/opt/resmon/bin/resls`. For additional information, refer to the man page for `resls(1m)`.

## Using the EMS HA Monitors

The EMS HA Monitors, available as a separate product (B5736DA), can be used to set up monitoring of disks and other resources as package dependencies. Examples of resource attributes that can be monitored using EMS include the following:

- Logical volume status
- Physical volume status
- System load
- Number of users
- File system utilization
- LAN health

Once a monitor is configured as a package dependency, the monitor will notify the package manager if an event occurs showing that a resource is down. The package may then be failed over to an adoptive node.

The EMS HA Monitors can also be used to report monitored events to a target application such as OpenView IT/Operations for graphical display or for operator notification. Refer to the manual *Using High Availability Monitors* (B5736-90022) for additional information.

## 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 Serviceguard Manager or in the ASCII package configuration file that determine each behavior.

**Table 3-3 Package Failover Behavior**

| Switching Behavior | Options in Serviceguard Manager | Parameters in ASCII File |
|---|---|---|
| Package switches normally after detection of service, network, or EMS failure. Halt script runs before switch takes place. (Default) | • Package Failfast set to Disabled. (Default)<br>• Service Failfast set to Disabled for all services. (Default)<br>• Automatic Switching set to Enabled for the package. (Default) | • NODE_FAIL_FAST_ENABLED set to NO. (Default)<br>• SERVICE_FAIL_FAST_ENABLED set to NO for all services. (Default)<br>• AUTO_RUN set to YES for the package. (Default) |
| Package fails over to the node with the fewest active packages. | • Failover policy set to Minimum Package Node. | • FAILOVER_POLICY set to MIN_PACKAGE_NODE. |
| Package fails over to the node that is next on the list of nodes. (Default) | • Failover policy set to Configured Node. (Default) | • FAILOVER_POLICY set to CONFIGURED_NODE. (Default) |
| 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. | • Failback policy set to Automatic. | • FAILBACK_POLICY set to AUTOMATIC. |

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

| Switching Behavior | Options in Serviceguard Manager | Parameters in ASCII File |
|---|---|---|
| If desired, package must be manually returned to its primary node if it is running on a non-primary node. | • Failback policy set to Manual. (Default)<br>• Failover policy set to Configured Node. (Default) | • FAILBACK_POLICY set to MANUAL. (Default)<br>• FAILOVER_POLICY set to CONFIGURED_NODE. (Default) |
| 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. | • Service Failfast set to Enabled for a specific service<br>• Automatic Switching set to Enabled for all packages. | • SERVICE_FAIL_FAST_ENABLED set to YES for a specific service.<br>• AUTO_RUN set to YES for all packages. |
| 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. | • Service Failfast set to Enabled for *all* services.<br>• Automatic Switching set to Enabled for all packages. | • SERVICE_FAIL_FAST_ENABLED set to YES for *all* services.<br>• AUTO_RUN set to YES for all packages. |

# 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. There can be a maximum of 150 packages per cluster and a total of 900 services per cluster.

## 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.

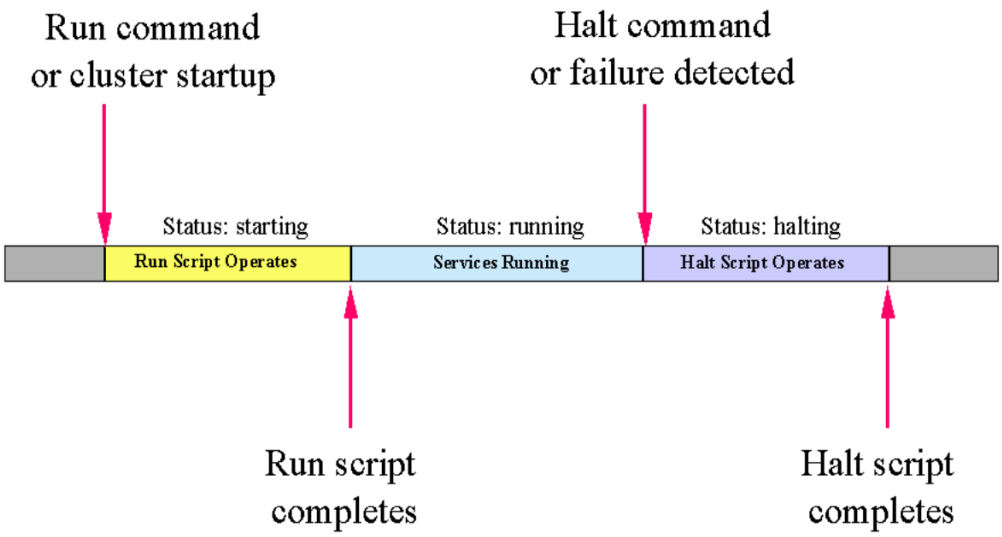
The package manager will always try to keep the package running unless there is something preventing it from running on any node. The most common reasons for a package not being able to run are that `AUTO_RUN` is disabled, or `NODE_SWITCHING` is disabled for the package on particular nodes. When a package 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.

**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 3-13.

**Figure 3-13 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, subnet, or monitored resource 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.

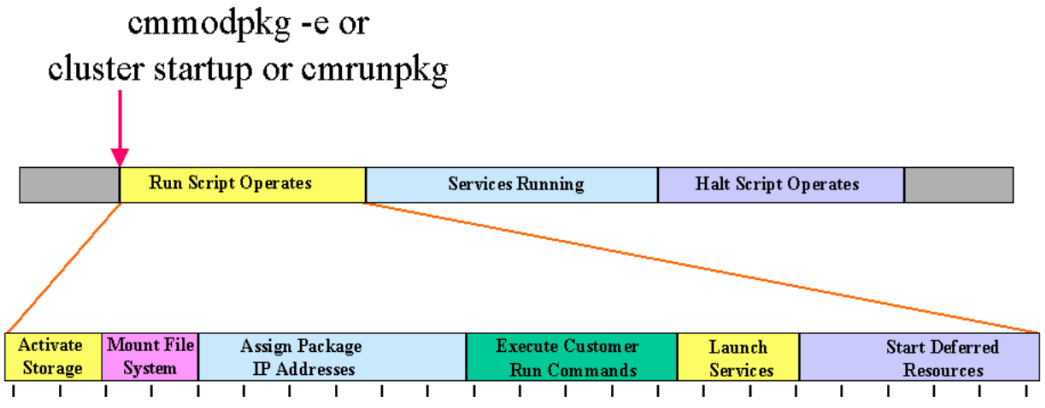
Once a node is selected, a check is then done to make sure the node allows the package to start on it. Then services are started up for a package by the control script on the selected node. Strictly speaking, the run script on the selected node is used to start the package.

## 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 3-14):

1. Activates volume groups or disk 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. Starts up any resources needed by the package that were specially marked for deferred startup.
7. Exits with an exit code of zero (0).

**Figure 3-14 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.

### While Services are Running

During the normal operation of cluster services, the package manager continuously monitors the following:

- Process IDs of the services
- Subnets configured for monitoring in the package configuration file
- Configured resources on which the package depends

Some failures can result in a local switch. For example, if there is a failure on a specific LAN card and there is a standby LAN configured for that subnet, then the Network Manager will switch to the healthy LAN

card. 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.

If there is a configured EMS resource dependency and there is a trigger that causes an event, the package will be halted.

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, Subnet, or Monitored Resource Fails

What happens when something goes wrong? If a service fails and there are no more restarts, if a subnet fails and there are no standbys, or if a configured resource 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.

**NOTE** If a package is dependent on a subnet, and the subnet on the primary node fails, then the package will start to shut down. If the subnet recovers immediately (before the package is restarted on an adoptive node), then the package could be restarted on the primary node. Therefore the package does not switch to another node in the cluster in this case.

## 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.

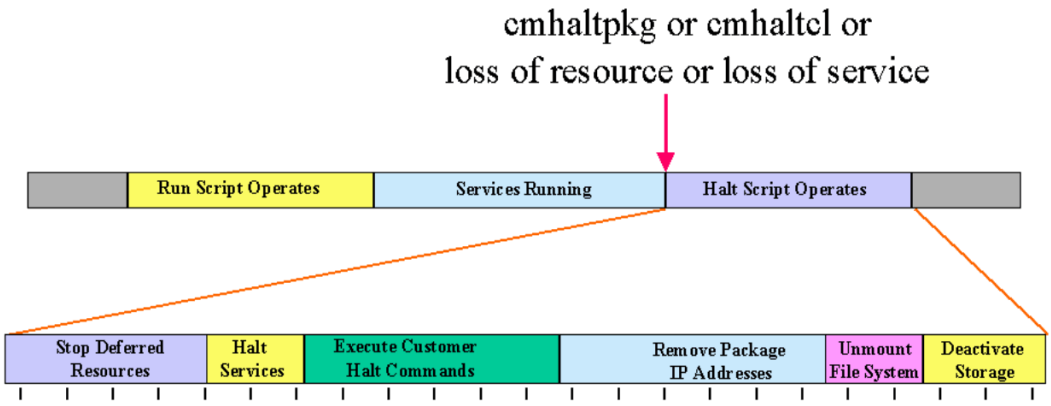
The cmmodpkg command cannot be used to halt a package, but it can disable switching either on particular nodes or on all nodes. A package can continue running when its switching has been disabled, but it will not be able to start on other nodes if it stops running on its current node.

## 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 3-15):

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

**Figure 3-15 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** | **HP-UX 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 |
| 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 |

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

| Package Error Condition | | | Results | | | |
|---|---|---|---|---|---|---|
| **Error or Exit Code** | **Node Failfast Enabled** | **Service Failfast Enabled** | **HP-UX 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 Exit 1 | NO | Either Setting | Running | N/A | Yes | No |
| Halt Script Timeout | YES | Either Setting | TOC | N/A | N/A (TOC) | Yes, unless the timeout happened after the cmhaltpkg 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 |
| Loss of Monitored Resource | YES | Either Setting | TOC | No | N/A (TOC) | Yes |

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

| Package Error Condition | | | Results | | | |
|---|---|---|---|---|---|---|
| **Error or Exit Code** | **Node Failfast Enabled** | **Service Failfast Enabled** | **HP-UX 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** |
| Loss of Monitored Resource | NO | Either Setting | Running | Yes | Yes, if the resource is not a deferred resource. No, if the resource is deferred. | 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, switching them when necessary.

## Stationary and Relocatable IP Addresses

Each node (host system) should have at least one IP address for each active network interface. This address, known as a **stationary IP address**, is configured in the node's `/etc/rc.config.d/netconf` file or in the node's `/etc/rc.config.d/netconf-ipv6` file. A stationary IP address is not transferable to another node, but may be transferable to a standby LAN interface card. The stationary IP address is *not* associated with packages. 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 **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. This can be a combination of IPv4 and IPv6 addresses.

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()`, which will return the package's relocatable IP address.

Both stationary and relocatable IP addresses will switch to a standby LAN interface in the event of a LAN card failure. In addition, 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.

### Types of IP Addresses

Both IPv4 and IPv6 address types are supported in Serviceguard. IPv4 addresses are the traditional addresses of the form “n.n.n.n” where ‘n’ is a decimal digit between 0 and 255. IPv6 addresses have the form “x:x:x:x:x:x:x:x” where ‘x’ is the hexadecimal value of each of eight 16-bit pieces of the 128-bit address. Only IPv4 addresses are supported as heartbeat addresses, but both IPv4 and IPv6 addresses (including various combinations) can be defined as stationary IPs in a cluster. Both IPv4 and IPv6 addresses also can be used as relocatable (package) IP addresses.

## 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 `cmmodnet` command in the **package control script**, which is described in detail in Chapter 6, “Configuring Packages and Their Services,” on page 243

IP addresses are configured only on each primary network interface card; standby cards are not configured with an IP address. Multiple IPv4 addresses on the same network card must belong to the same IP subnet.

### Load Sharing

In one package, it is possible to have multiple services that are associated with the same IP address. If one service is moved to a new system, then the other services using the IP address will also be migrated. Load sharing can be achieved by making each service its own package and giving it a unique IP address. This gives the administrator the ability to move selected services to less loaded systems.

## Monitoring LAN Interfaces and Detecting Failure

At regular intervals, Serviceguard polls all the network interface cards specified in the cluster configuration file. Network failures are detected within each single node in the following manner. One interface on the

node is assigned to be the poller. The poller will poll the other primary and standby interfaces in the same bridged net on that node to see whether they are still healthy. Normally, the poller is a standby interface; if there are no standby interfaces in a bridged net, the primary interface is assigned the polling task. (Bridged nets are explained in the section on "Redundant Network Components" in Chapter 2.)

The polling interface sends LAN packets to all other interfaces in the node that are on the same bridged net and receives packets back from them.

Whenever a LAN driver reports an error, Serviceguard immediately declares that the card is bad and performs a local switch, if applicable. For example, when the card fails to send, Serviceguard will immediately receive an error notification and it will mark the card as down.

Serviceguard Network Manager also looks at the numerical counts of packages and received on an interface to determine if a card is having a problem. There are two ways Serviceguard can handle the counts of packets sent and received. In the cluster configuration file, choose one of these two values for the NETWORK_FAILURE_DETECTION parameter:

- INOUT: When both the inbound and outbound counts stop incrementing for a predetermined amount of time, Serviceguard will declare the card as bad. Serviceguard will not declare the card as bad if only the inbound or only the outbound count stops incrementing. *Both* must stop. This is the default.
- INONLY_OR_INOUT: This option will also declare the card as bad if both inbound and outbound counts stop incrementing. However, it will also declare it as bad if only the inbound count stops.

  This option is new with Serviceguard A.11.16. It is not suitable for all environments. Before choosing it, be sure these conditions are met:

  — All bridged nets in the cluster should have more than two interfaces each.

  — Each primary interface should have at least one standby interface, and it should be connected to a standby switch.

  — The primary switch should be directly connected to its standby.

  — There should be no single point of failure anywhere on all bridged nets.

### Local Switching

A local network switch involves the detection of a local network interface failure and a failover to the local backup LAN card. The backup LAN card must not have any IP addresses configured.

In the case of local network switch, TCP/IP connections are not lost for Ethernet, but IEEE 802.3 connections will be lost. For IPv4, Ethernet, Token Ring and FDDI use the ARP protocol, and HP-UX sends out an unsolicited ARP to notify remote systems of address mapping between MAC (link level) addresses and IP level addresses. IEEE 802.3 does not have the *rearp* function.

IPv6 uses the Neighbor Discovery Protocol (NDP) instead of ARP. The NDP protocol is used by hosts and routers to do the following:

- determine the link-layer addresses of neighbors on the same link, and quickly purge cached values that become invalid.
- find neighboring routers willing to forward packets on their behalf.
- actively keep track of which neighbors are reachable, and which are not, and detect changed link-layer addresses.
- search for alternate functioning routers when the path to a router fails.

Within the Ethernet family, local switching configuration is supported:

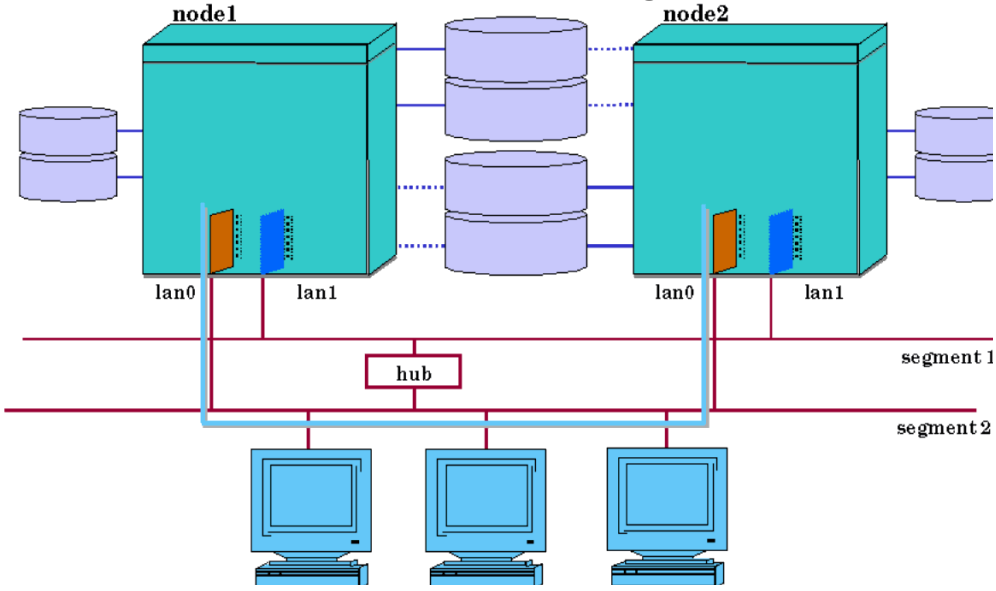
- 1000Base-SX and 1000Base-T
- 1000Base-T or 1000BaseSX and 100Base-T

On HP-UX 11i, however, Jumbo Frames must not be used since the 100Base-T cards do not support Jumbo Frames. Network interface cards running 1000Base-T or 1000Base-SX cannot do local failover to 10BaseT.

During the transfer, IP packets will be lost, but TCP (Transmission Control Protocol) will retransmit the packets. In the case of UDP (User Datagram Protocol), the packets will not be retransmitted automatically by the protocol. However, since UDP is an unreliable service, UDP applications should be prepared to handle the case of lost network packets and recover appropriately. Note that a local switchover is supported only between two LANs of the same type. For example, a local switchover between Ethernet and FDDI interfaces is not supported, but a local switchover between 10BT Ethernet and 100BT Ethernet is supported.

Figure 3-16 shows two nodes connected in one bridged net. LAN segments 1 and 2 are connected by a hub.

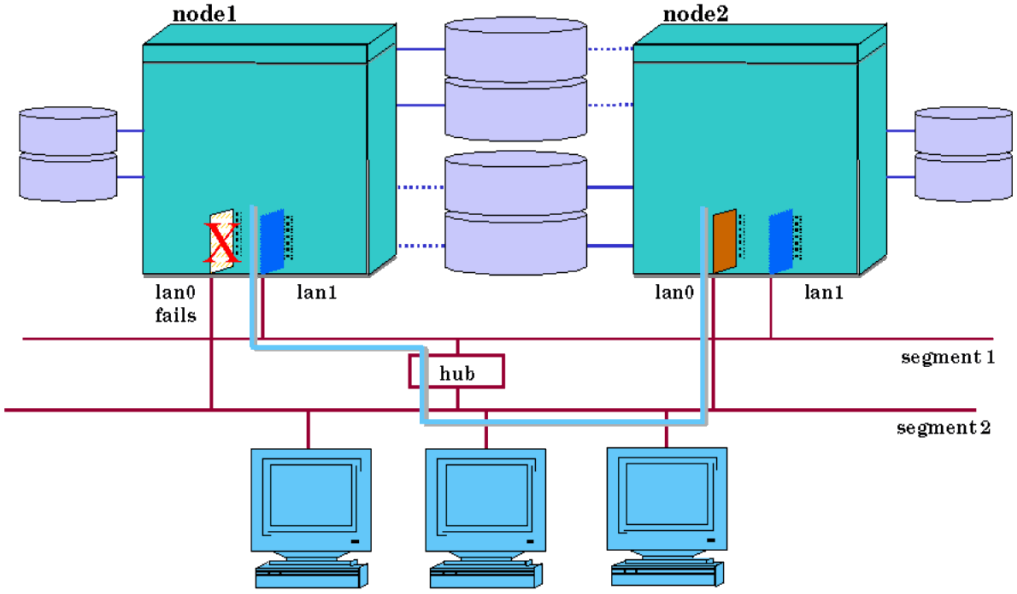
**Figure 3-16 Cluster Before Local Network Switching**




Node 1 and Node 2 are communicating over LAN segment 2. LAN segment 1 is a standby.

In Figure 3-17, we see what would happen if the LAN segment 2 network interface card on Node 1 were to fail.

**Figure 3-17 Cluster After Local Network Switching**

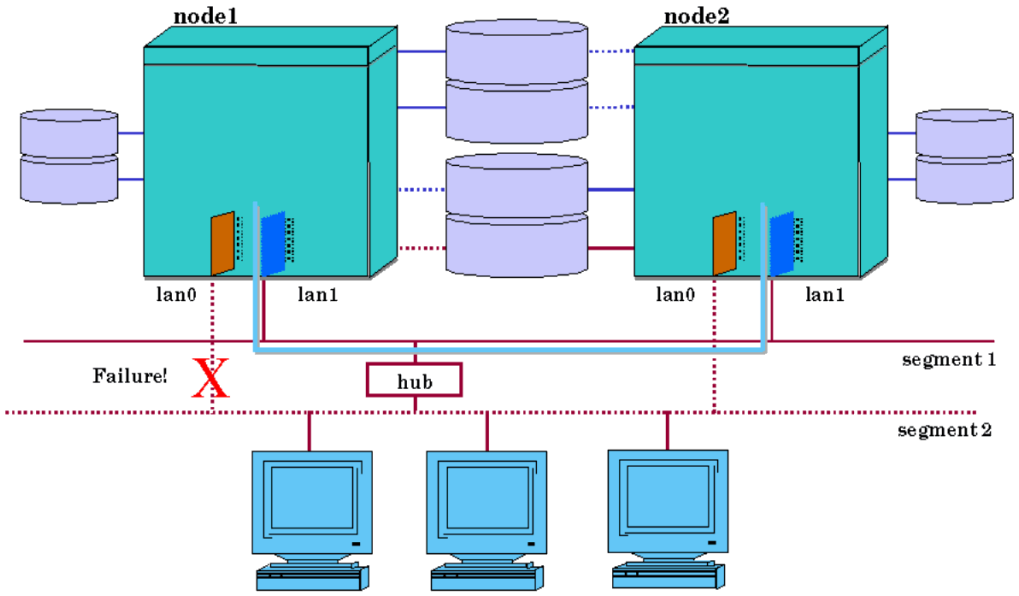



As the standby interface takes over, IP addresses will be switched to the hardware path associated with the standby interface. The switch is transparent at the TCP/IP level. All applications continue to run on their original nodes. During this time, IP traffic on Node 1 will be delayed as the transfer occurs. However, the TCP/IP connections will continue to be maintained and applications will continue to run. Control of the packages on Node 1 is not affected.

**NOTE** On Ethernet networks, Serviceguard supports local failover between network interfaces configured with “Ethernet protocol” or between network interfaces configured with “SNAP encapsulation within IEEE 802.3 protocol.” You cannot use both protocols on the same interface, nor can you have a local failover between interfaces that are using different protocols.

Another example of local switching is shown in Figure 3-18. In this case a failure affecting segment 2 causes both nodes to switch to the LAN cards attached to segment 1.

**Figure 3-18 Local Switching After Cable Failure**




Local network switching will work with a cluster containing one or more nodes. You may wish to design a single-node cluster in order to take advantages of this local network switching feature in situations where you need only one node and do not wish to set up a more complex cluster.

### Switching Back to Primary LAN Interfaces after Local Switching

Whenever a node is halted, the cluster daemon (`cmcld`) will always attempt to switch any Serviceguard-configured subnets running on standby interfaces back to the primary interfaces. This is done regardless of the link state of the primary interfaces. The intent of this switchback is to preserve the original network configuration as it was before the cluster started. Switching back occurs on the specified node if a `cmhaltnode` command is issued or on all nodes in the cluster if a `cmhaltcl` command is issued.

### 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.

Note that remote switching is supported only between LANs of the same type. For example, a remote switchover between Ethernet on one machine and FDDI interfaces on the failover machine is not supported. The remote switching of relocatable IP addresses was shown previously in Figure 3-5 and Figure 3-6.

### Address Resolution Messages after Switching

When a floating IPv4 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 IPv4 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.

Unlike IPv4, IPv6 addresses use NDP messages to determine the link-layer addresses of its neighbors.

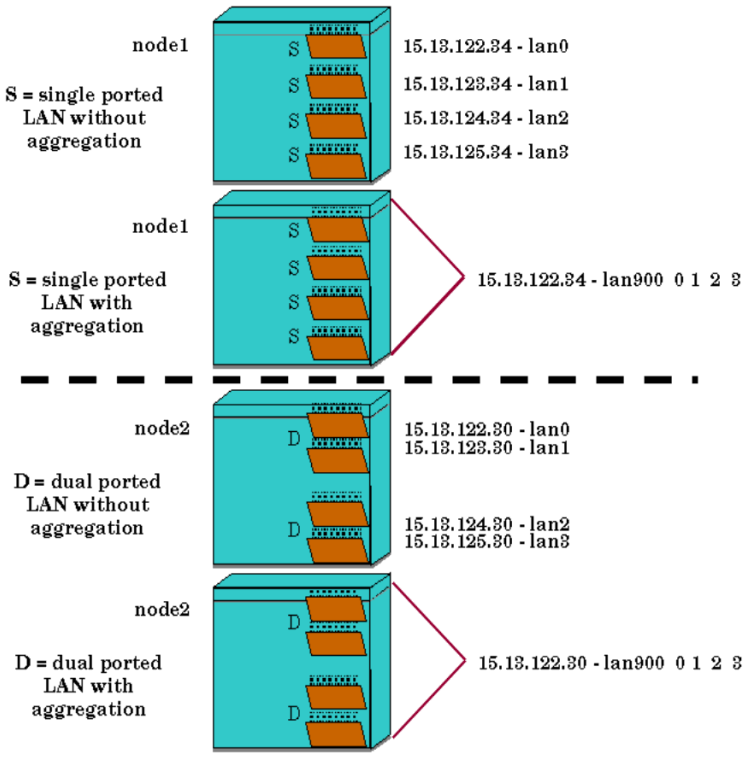
## Automatic Port Aggregation

Serviceguard supports the use of automatic port aggregation through HP-APA (Auto-Port Aggregation, HP product J4240AA). HP-APA is a networking technology that aggregates multiple physical Fast Ethernet or multiple physical Gigabit Ethernet ports into a logical link aggregate. HP-APA allows a flexible, scalable bandwidth based on multiple 100 Mbps Fast Ethernet links or multiple 1 Gbps Ethernet links (or 200 Mbps and 2 Gbps full duplex respectively). Its other benefits include load balancing between physical links, automatic fault detection, and recovery for environments which require high availability. Port aggregation capability is sometimes referred to as link aggregation or trunking. APA is also supported on dual-stack kernel.

Once enabled, each link aggregate can be viewed as a single logical link of multiple physical ports with only one IP and MAC address. HP-APA can aggregate up to four physical ports into one link aggregate; the number of link aggregates allowed per system is 50. Empty link aggregates will have zero MAC addresses.

You can aggregate the ports within a multi-ported networking card (cards with up to four ports are currently available). Alternatively, you can aggregate ports from different cards. Figure 3-19 shows two examples.

**Figure 3-19 Aggregated Networking Ports**




Both the Single and Dual ported LANs in the non-aggregated configuration have four LAN cards, each associated with a separate non-aggregated IP address and MAC address, and each with its own LAN name (lan0, lan1, lan2, lan3). 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 lan900, the name by which the aggregate is known on HP-UX 11i (on HP-UX 11.0, the aggregates would begin with lan100).

Various combinations of Ethernet card types (single or dual-ported) and aggregation groups are possible, but it is vitally important to remember that at least two physical cards must be used in any combination of APA's to avoid a single point of failure for heartbeat connections. HP-APA currently supports both automatic and manual configuration of link aggregates.

For information about implementing APA with Serviceguard, see *HP Auto Port Aggregation (APA) Support Guide* and other APA documents posted at *http://docs.hp.com* in the Networking and Communications collection.

## VLAN Configurations

Virtual LAN configuration using HP-UX VLAN software is now supported in Serviceguard clusters. VLAN is also supported on dual-stack kernel.

### What is VLAN?

Virtual LAN (or VLAN) is a networking technology that allows grouping of network nodes based on an association rule regardless of their physical locations. Specifically, VLAN can be used to divide a physical LAN into multiple logical LAN segments or broadcast domains. Each interface in a logical LAN will be assigned a tag id at the time it is configured. VLAN interfaces, which share the same tag id, can communicate to each other as if they were on the same physical network. The advantages of creating virtual LANs are to reduce broadcast traffic, increase network performance and security, and improve manageability. On HP-UX, initial VLAN association rules are port-based, IP-based, and protocol-based. Multiple VLAN interfaces can be configured from a physical LAN interface and then appear to applications as regular network interfaces. IP addresses can then be assigned on these VLAN interfaces to form their own subnets. Please refer to the document *Using HP-UX VLAN* (T1453-90001) for more details on how to configure VLAN interfaces.

### Support for HP-UX VLAN

VLAN is supported with Serviceguard starting from A.11.14 on HP-UX 11i.

The support of VLAN is similar to other link technologies. VLAN interfaces can be used as heartbeat as well as data networks in the cluster. The Network Manager will monitor the health of VLAN interfaces configured in the cluster, and perform local and remote failover of VLAN interfaces when failure is detected. The failure of VLAN interfaces typically occurs when the physical NIC port, upon which they are created, has failed.

### Configuration Restrictions

HP-UX allows up to 1024 VLANs to be created from a physical NIC port. Obviously, a large pool of system resources is required to accommodate such a configuration. With the availability of VLAN technology, Serviceguard may face potential performance degradation, high CPU utilization and memory shortage issues if there is a large number of network interfaces configured in each cluster node. To provide enough flexibility in VLAN networking, Serviceguard solutions should adhere to the following VLAN and general network configuration requirements:

- A maximum of 30 network interfaces per node is supported in the cluster ASCII file. The interfaces can be physical NIC ports, VLAN interfaces, APA aggregates, or any combination of these.
- The physical LAN interfaces, upon which the VLAN interfaces are created, should be configured in the ASCII file along with the VLAN interfaces to meet Serviceguard heartbeat requirements.
- A maximum of 14 VLAN interfaces per physical NIC port is supported if there are at least two NIC ports configured with VLAN in a cluster node. This limit can be increased to 28 VLAN interfaces if the physical NIC port is the only NIC port configured with VLAN. These restrictions are bound by the 30 network interfaces per node limit. One example to show this configuration is a cluster node with 2 single-ported NIC cards, both of which are configured as heartbeat networks. Then each can have up to 14 VLAN interfaces per NIC card totaling 30 network interfaces per node.
- Local failover of VLANs must be on same link types. The primary and standby VLANs must have same vlan id (or tag id).
- VLAN configurations are only supported on HP-UX 11i.

- Only port-based and IP subnet-based VLANs are supported. Protocol-based VLAN will not be supported because Serviceguard does not support any transport protocols other than TCP/IP.
- Each VLAN interface must be assigned an IP address in a unique subnet in order to operate properly unless it is used as a standby of a primary VLAN interface.
- VLAN interfaces over APA aggregates are not supported.
- Failover from physical LAN interface to VLAN interfaces or vice versa is not supported due to restrictions of VLAN software.
- Using VLAN in a Wide Area Network cluster is not supported.

**Enhanced Heartbeat Requirements**

VLAN technology allows greater flexibility in network configurations in the enterprise. In order to allow Serviceguard to be running on such dynamic environments successfully while maintaining its reliability and availability features simultaneously, the existing heartbeat rules must continue to be applied with more restrictions when VLAN interfaces are present in the cluster:

1. The existing minimum heartbeat requirements are unchanged, but they will be checked against the physical networks instead of any LAN interfaces, which may include VLAN interfaces, to avoid single points of failure.
2. Heartbeats are still recommended on all networks, including VLANs.
3. If VLANs are not preferred for heartbeat networks, then heartbeats are recommended for all remaining physical networks present in the cluster ASCII file.

# Volume Managers for Data Storage

A volume manager is a tool that lets you create units of disk storage known as storage groups. Storage groups contain logical volumes for use on single systems and in high availability clusters. In Serviceguard clusters, storage groups are activated by package control scripts.

## Types of Redundant Storage

In Serviceguard, there are two types of supported shared data storage: mirrored **individual disks** (also known as JBODs, for "just a bunch of disks"), and external **disk arrays** which configure redundant storage in hardware. Mirroring is known as RAID1, while the redundancy available in a disk array is known as RAID5 (or similar). Here are some differences between the two storage methods:

- If you are using JBODs, the basic element of storage is an individual disk. This disk must be paired with another disk to create a mirror (RAID1). (Serviceguard configurations usually have separate mirrors on different storage devices).
- If you have a disk array, the basic element of storage is a LUN, which already provides storage redundancy via RAID5.

## Examples of Mirrored Storage

Figure 3-20 shows an illustration of mirrored storage using HA storage racks. In the example, node1 and node2 are cabled in a parallel configuration, each with redundant paths to two shared storage devices. Each of two nodes also has two (non-shared) internal disks which are used for the root file system, swap etc. Each shared storage unit has three disks, The device file names of the three disks on one of the two storage units are `c0t0d0`, `c0t1d0`, and c0t2d0. On the other, they are c1t0d0, `c1t1d0`, and c1t2d0.

**Figure 3-20 Physical Disks Within Shared Storage Units**

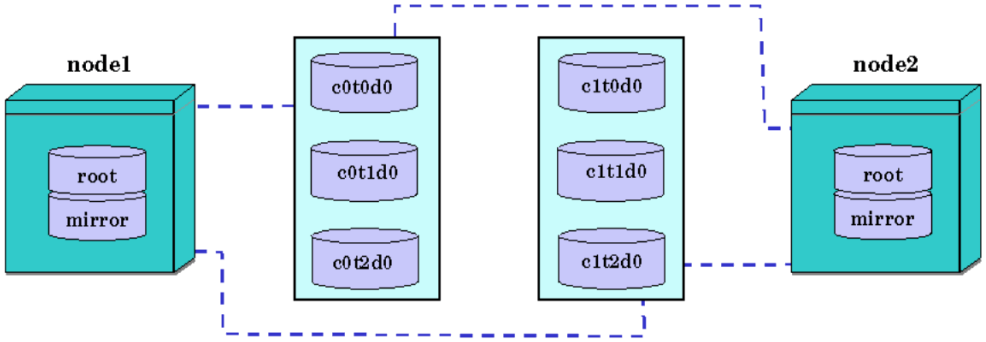



Figure 3-21 shows the individual disks combined in a multiple disk mirrored configuration.

**Figure 3-21 Mirrored Physical Disks**

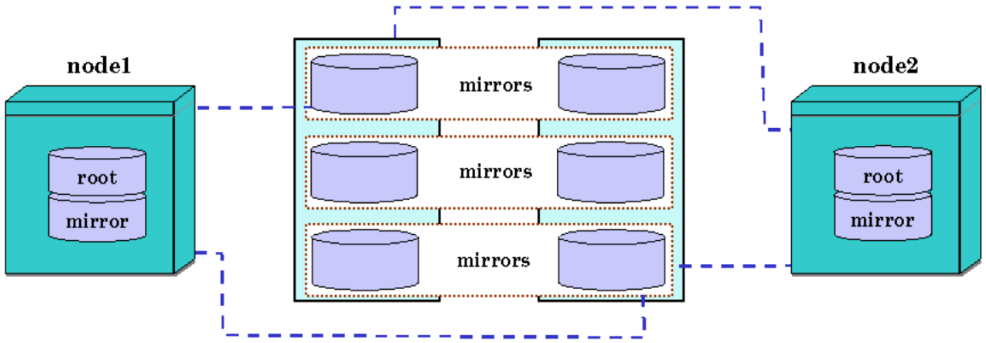
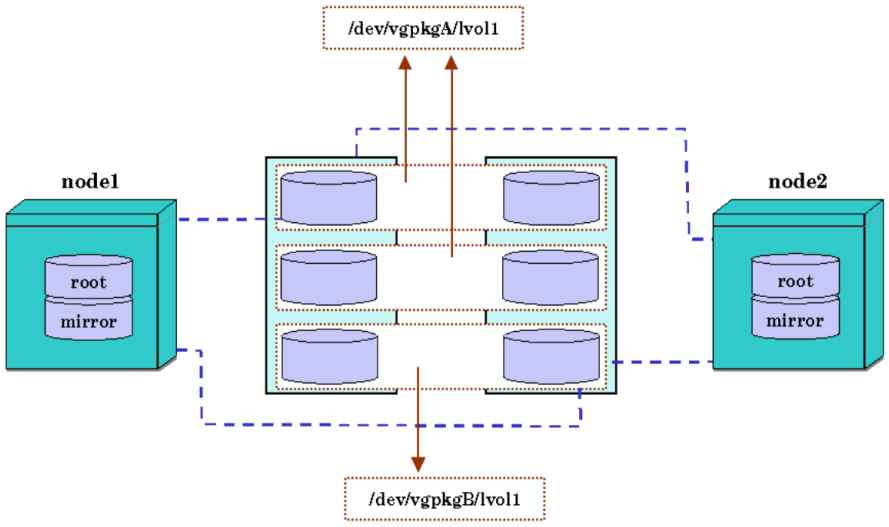



Figure 3-22 shows the mirrors configured into LVM volume groups, shown in the figure as `/dev/vgpkgA` and `/dev/vgpkgB`. The volume groups are activated by Serviceguard packages for use by highly available applications.

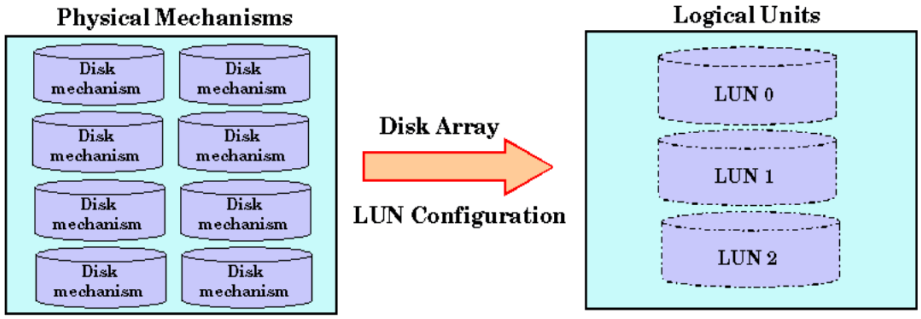
**Figure 3-22 Multiple Devices Configured in Volume Groups**




## Examples of Storage on Disk Arrays

Figure 3-23 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.

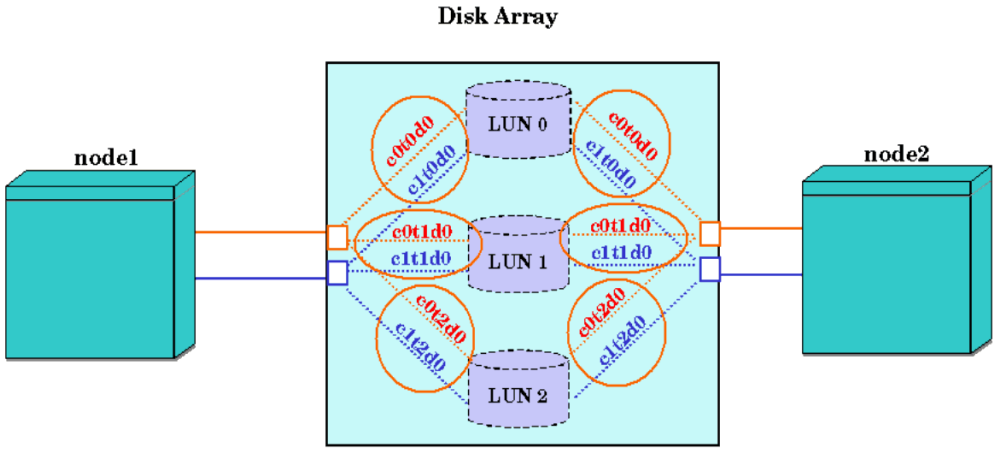
**Figure 3-23 Physical Disks Combined into LUNs**



**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.

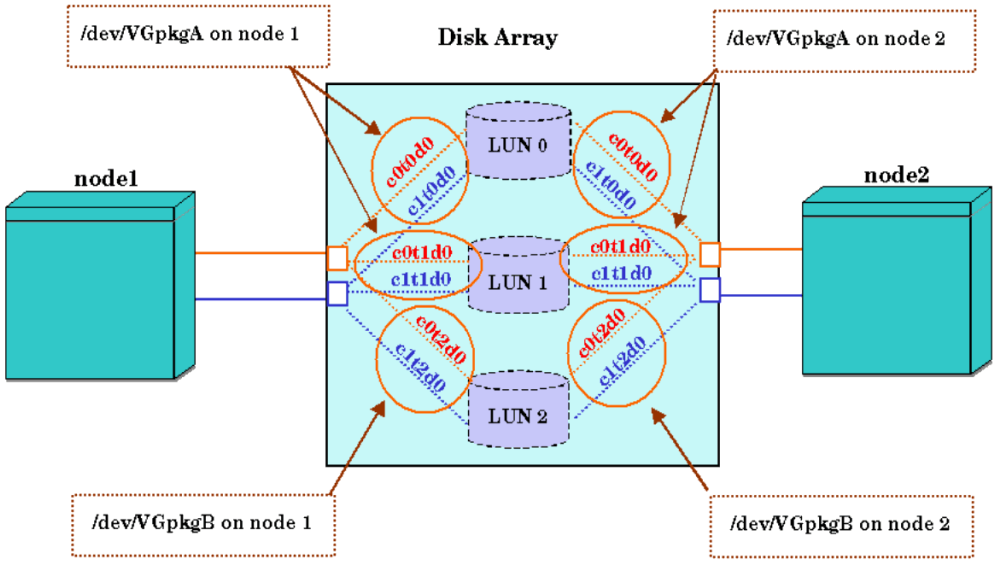
Figure 3-24 shows LUNs configured with multiple paths (links) to provide redundant pathways to the data.

**Figure 3-24 Multiple Paths to LUNs**




Finally, the multiple paths are configured into volume groups as shown in Figure 3-25.

**Figure 3-25 Multiple Paths in Volume Groups**




## Types of Volume Manager

Serviceguard allows a choice of volume managers for data storage:

- HP-UX Logical Volume Manager (LVM) and (optionally) MirrorDisk/UX
- VERITAS Volume Manager for HP-UX (VxVM)—Base and Add-on Products
- VERITAS Cluster Volume Manager for HP-UX (CVM)

Separate sections in Chapters 5 and 6 explain how to configure cluster storage using all of these volume managers. The rest of the present section explains some of the differences among these available volume managers and offers suggestions about appropriate choices for your cluster environment.

**NOTE** The HP-UX Logical Volume Manager is described in *Managing Systems and Workgroups*. A complete description of VERITAS volume management products is available in the *VERITAS Volume Manager for HP-UX Release Notes*.

## HP-UX Logical Volume Manager (LVM)

Logical Volume Manager (LVM) is the legacy storage management product on HP-UX. Included with the operating system, LVM is available on all cluster nodes. It supports the use of MirrorDisk/UX, which is an add-on product that allows disk mirroring with up to two mirrors (for a total of three copies of the data).

Currently, the HP-UX root disk can be configured as an LVM volume group. (Note that, in this case, the HP-UX root disk is not the same as the VERITAS *root disk group*, rootdg, which must be configured in addition to the HP-UX root disk on any node that uses VERITAS Volume Manager products.) The Serviceguard cluster lock disk also is configured using a disk configured in an LVM volume group.

LVM continues to be supported on HP-UX single systems and on Serviceguard clusters.

## VERITAS Volume Manager (VxVM)

The Base VERITAS Volume Manager for HP-UX (Base-VXVM) is provided at no additional cost with HP-UX 11i. This includes basic volume manager features, including a Java-based GUI, known as `vmsa`. It is possible to configure cluster storage for Serviceguard with only Base-VXVM. However, only a limited set of features is available.

The add-on product, VERITAS Volume Manager for HP-UX (B9116AA), provides a full set of enhanced volume manager capabilities in addition to basic volume management. This includes features such as mirroring, dynamic multipathing for active/active storage devices, and hot relocation. The VERITAS FastResync Option for HP-UX (B9118AA) is another add-on product, available with an additional charge, that reduces the time taken to resynchronize a split mirror to the volume. This product requires the purchase of VERITAS Volume Manager for HP-UX (B9116AA).

VxVM can be used in clusters that:

- are of any size, up to 16 nodes.
- require a fast cluster startup time.
- do not require shared storage group activation.
- do not have all nodes cabled to all disks.
- need to use RAID 5 or striped mirroring.
- have multiple heartbeat subnets configured.

### Propagation of Disk Groups in VxVM

With VxVM, a disk group can be created on any node, with the cluster up or not. The user then needs to go to each node and validate that disk group by trying to import it. Thus, although there are more steps required for propagating disk groups with VxVM than with CVM, you have the freedom to create the disk group from any node.

### Package Startup Time with VxVM

With VxVM, each disk group is imported by the package control script that uses the disk group. This means that cluster startup time is not affected, but individual package startup time might be increased because VxVM imports the disk group at the time the package starts up.

## VERITAS Cluster Volume Manager (CVM)

You may choose to configure cluster storage with the VERITAS Cluster Volume Manager (CVM) instead of the Volume Manager (VxVM). The Base-VXVM provides some basic cluster features when Serviceguard is installed; up to four nodes are supported, but there is no support for software mirroring, dynamic multipathing (for active/active storage devices), or numerous other features that require the additional licenses. The VERITAS Cluster Volume Manager, CVM (B9117AA) is an enhanced version of the VxVM volume manager that is specifically designed for cluster use. When installed with the VERITAS Volume Manager (B9116AA), the CVM add-on product provides most of the enhanced VxVM features in a clustered environment. CVM is truly cluster-aware, obtaining information about cluster membership from Serviceguard directly. Cluster information is provided via a special

system multi-node package known as VxVM-CVM-pkg, which runs on all nodes in the cluster. The cluster must be up and must be running this package in order to configure VxVM disk groups for use with CVM.

CVM allows you to activate storage on one node at a time, or you can perform write activation on one node and read activation on another node at the same time (for example, allowing backups). CVM provides full mirroring and dynamic multipathing (DMP) for clusters.

CVM can be used in clusters that:

- will run applications that require fast disk group activation after package failover.
- require activation on more than one node at a time, for example to perform a backup from one node while a package using the volume is active on another node. In this case, the package using the disk group would have the disk group active in exclusive write mode while the node that is doing the backup would have the disk group active in shared read mode.
- are configured with only a single heartbeat subnet.

CVM is supported on 4 nodes or fewer at this release. Shared storage devices must be connected to all nodes in the cluster, whether or not the node accesses data on the device.

### Cluster Startup Time with CVM

With CVM, all shared disk groups (DGs) are imported when the VxVM-CVM-pkg control script completes. Depending on the number of DGs, the number of nodes and the configuration of these (number of disks, volumes, etc.) this can take some time (current timeout value for this package is 3 minutes but for larger configurations this may have to be increased). Any failover package that uses a CVM DG will not start until the VxVM-CVM-pkg is up. Note that this delay does not affect package failover time; it is a one-time overhead cost at cluster startup.

### Propagation of Disk Groups with CVM

With CVM, disk groups are created on one cluster node known as the CVM master node. The cluster has to be running to create shared disk groups. CVM will validate that each node can see each disk and will not allow invalid DGs to be created.

### Single Heartbeat Subnet Required with CVM

It is normally recommended that you configure all subnets that interconnect cluster nodes as heartbeat networks, since this increases protection against multiple faults at no additional cost. However, if you will be using the VERITAS Cluster Volume Manager (CVM), only a single heartbeat subnet is allowed. When the VxVM-CVM-pkg is added (as shown in Chapter 5), `cmcheckconf` and `cmapplyconf` will check to ensure that only a single heartbeat subnet is configured. If more heartbeat subnets are configured, the `cmcheckconf` and `cmapplyconf` will fail.

This restriction is due to the design of the VERITAS Cluster Daemon (`vxclustd`), which is started within the VxVM-CVM-pkg. This daemon is only able to communicate information to cluster nodes on a single IP connection on a single subnet. Moreover, communication between nodes in the cluster must be the same for the Veritas Cluster Daemon as for the Serviceguard cluster daemon `cmcld`. Therefore, cmcld must also only have the use of a single heartbeat subnet.

There are tradeoffs in only using a single subnet. If the heartbeat is not heard for more than a `NODE_TIMEOUT` interval, the cluster will re-form. Depending on the length of the disruption, this could cause one or all of the cluster nodes to TOC. This is most commonly seen when there is heavy traffic on the subnet preventing the heartbeat packets from being transmitted. This can also been seen when there are network configuration problems or network hardware problems that are not easily recovered and may not be detected by Serviceguard or APA. The most common way to avoid problems in these scenarios is to utilize multiple heartbeat networks.

In the case of CVM, the single heartbeat subnet should be configured with standby LANS or as a group of aggregated ports to help minimize the possibility of system or cluster failures.

## Comparison of Volume Managers

The following table summarizes some of the advantages and disadvantages of the volume managers that are currently available.

**Table 3-5 Pros and Cons of Volume Managers with Serviceguard**

| Product | Pros | Cons |
|---|---|---|
| Logical Volume Manager (LVM) | • Legacy system is robust and familiar to HP-UX users<br>• Existing packages do not need to be changed<br>• Supports up to 16 nodes per cluster<br>• Supports use of PV links for multiple data paths<br>• Supports exclusive activation as well as read-only activation from multiple nodes<br>• Can be used to configure a cluster lock disk<br>• Supports multiple heartbeat subnets | • Lacks flexibility and extended features of other volume managers<br>• PV links are active/standby only, with one link active at a time |
| Base-VxVM | • Software is free with HP-UX 11i and later<br>• Simple basic features provided<br>• Supports multiple heartbeat subnets | • Limited feature set<br>• Cannot be used as a cluster lock device |

**Table 3-5 Pros and Cons of Volume Managers with Serviceguard**

| Product | Pros | Cons |
|---|---|---|
| VERITAS Volume Manager—B9116AA (VxVM) | • Disk group configuration from any node<br>• Enhanced set of volume management features, including software mirroring, RAID 0/1, RAID 5, and dynamic multi-pathing for active/active storage devices<br>• Cluster startup time is faster than with CVM.<br>• Supports up to 16 nodes per cluster<br>• Supports multiple heartbeat subnets | • Does not support activation on multiple nodes in either shared mode or read-only mode<br>• May cause delay at package startup time due to lengthy vxdg import<br>• Enhanced features require purchase of additional license<br>• Cannot be used as a cluster lock device |
| VERITAS Cluster Volume Manager—B9117AA (CVM) | • Enhanced set of volume management features, including software mirroring and RAID 0/1<br>• Package startup time is faster than with VxVM<br>• Supports exclusive activation<br>• Supports activation in different modes on different nodes at the same time | • Disk groups must be configured on a master node<br>• Can only be used with up to 4 cluster nodes<br>• Cluster startup may be slower than with VxVM<br>• Requires purchase of additional license<br>• No support for striped mirrors or RAID 5<br>• Supports only a single heartbeat subnet |

# Responses to Failures

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 an Serviceguard cluster is an HP-UX 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 predetermined time, or if there is a kernel hang, a kernel spin, a runaway real-time process, or if the Serviceguard cluster daemon, *cmcld*, fails. During this event, a system dump is performed and the following message is sent to the console:

```
Serviceguard: Unable to maintain contact with cmcld daemon.
Performing TOC to ensure data integrity.
```

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 (`cmcld`) or other critical daemons will cause a node to fail even when its packages and their services are functioning.

In a very few 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, a local switch is done to a standby LAN interface if one exists. If a heartbeat LAN interface fails and no standby or 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 protection is provided by separate products, such as MirrorDisk/UX in LVM or VERITAS mirroring in VxVM and CVM. In addition, separately available EMS disk monitors allow you to notify operations personnel when a specific failure, such as a lock disk failure, takes place. Refer to the manual *Using High Availablity Monitors* (HP part number B5736-90042) for additional information.

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 the package manager receives a report of an EMS monitor event showing that a configured resource dependency is not met, the package fails and tries to restart on the alternate node.

If you wish, you can modify this default behavior by specifying that the node should crash (TOC) before the transfer takes place. (In a very few cases, Serviceguard will attempt to reboot the system prior to a TOC when this behavior is specified.) 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. In a two-node cluster, the use of an RS-232 line prevents a TOC from the momentary loss of heartbeat on the LAN due to network saturation. The RS232 line also assists in quickly detecting network failures when they occur.

# 4 Planning and Documenting an HA Cluster

Building an 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 the graphical user interface, Serviceguard Manager, or a variety of HP-UX 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 input with Serviceguard Manager or edited into the ASCII configuration files and control scripts.

With Serviceguard version 11.16, the SAM high availability tool is no longer available. Configuration of Serviceguard clusters with version 11.16 or later is now done through Serviceguard Manager version A.04.00 or later. Administration of Serviceguard clusters with version 11.12 or later is now done with Serviceguard Manager version 11.12 or later. For more information about Serviceguard Manager, see Chapter 7. For information about using SAM on older versions, see earlier editions of this manual.

When the configuration is finished, you can use Serviceguard Manager to capture all the configuration data in a.sgm file. Later, you can see the information in Serviceguard Manager. Clusters and packages that are configured in Serviceguard Manager can be modified in Serviceguard Manager. 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
- LVM Planning

- CVM and VxVM 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 G.

**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 depends on the number of packages configured in the cluster. The following equation provides a rough estimate of how much memory is needed:

```
Total Memory = 6 MB + 100 KB/package
                          in cluster
```

The total amount is needed *on all nodes in the cluster*, regardless of whether a given package 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:

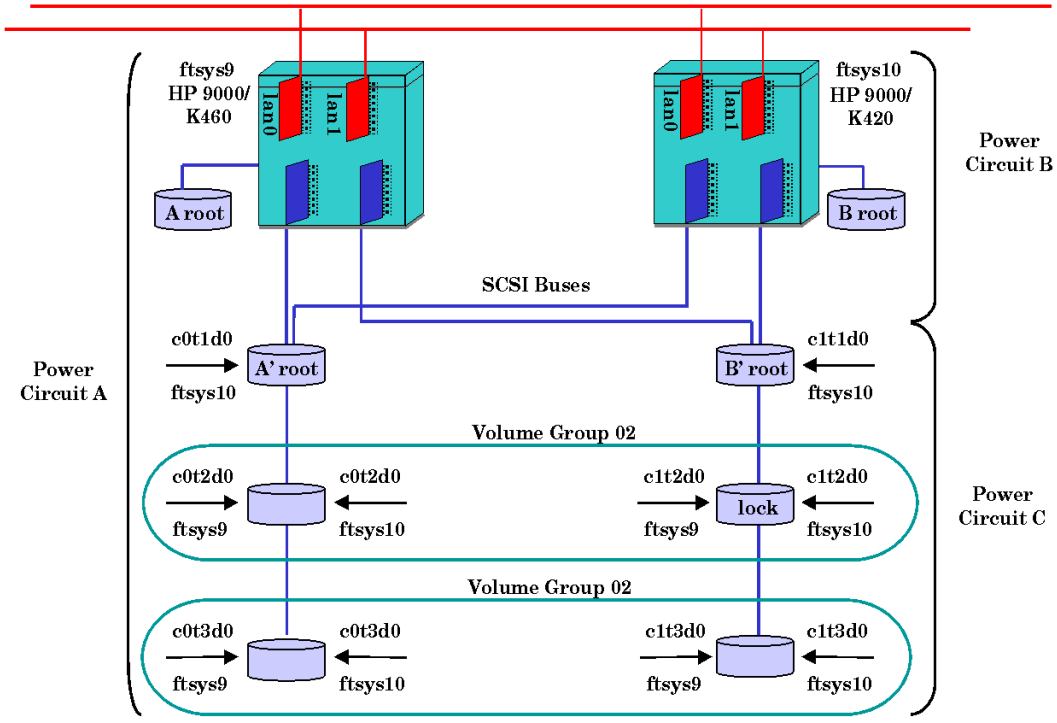
- Set the Maximum Configured Packages parameter (described later in this chapter in the "Cluster Configuration Planning" section) high enough to accommodate the additional packages you plan to add. Keep in mind that adding package capacity uses memory resources (100 KB per package).
- Plan SCSI bus cabling to allow the addition of more disk hardware for shared data storage if needed. You should attach inline SCSI terminator cables to SCSI ports that you intend to use for additional devices. This allows you to add them while the bus is still active.
- Remember the rules for cluster locks when considering expansion. A one-node cluster does not require a cluster lock. A two-node cluster must have a cluster lock. In clusters larger than 3 nodes, a cluster lock is strongly recommended. If you have a cluster with more than 4 nodes, you can use a quorum server but a cluster lock disk is not allowed.
- 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.
- Resources monitored by EMS should be pre-configured into the cluster configuration if they will be needed for packages you will add later while the cluster is running. Once a resource dependency is configured for any package in the cluster, it will always be available for later packages to use. However, you cannot add a never-before-configured resource to a package while the package 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. 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*. The form has three parts:

- SPU Information

- Network Information
- Disk I/O Information

## SPU Information

SPU information includes the basic characteristics of the systems you are using in the cluster. Different models of computers can be mixed in the same cluster. This configuration model also applies to HP Integrity servers. The series 700 workstations are not supported for Serviceguard.

On one worksheet per node, include the following items:

*Server Number*

Enter the series number; for example, `rp8400 or rx8620-32`.

*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.

## Network Information

Serviceguard monitors LAN interfaces as well as serial lines (RS232) configured to carry cluster heartbeat only.

**NOTE** Serviceguard does not support communication across routers between nodes in the same cluster.

Serviceguard communication relies on the exchange of DLPI (Data Link Provider Interface) traffic at the data link layer and the UDP/TCP (User Datagram Protocol/Transmission Control Protocol) traffic at the Transport layer between cluster nodes.

### LAN Information

While a minimum of one LAN interface per subnet is required, at least two LAN interfaces, one primary and one or more standby, 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 `lanscan` after you install the card.

*IP Address*

Enter this node's host IP address(es), to be used on this interface. If the interface is a standby and does not have an IP address, enter 'Standby.'

An IPv4 address is a string of 4 digits separated with decimals, in this form:

`nnn.nnn.nnn.nnn`

An IPV6 address is a string of 8 hexadecimal values separated with colons, in this form:

xxx:xxx:xxx:xxx:xxx:xxx:xxx:xxx.

For more details of IPv6 address format, see the Appendix , "IPv6 Address Types," on page 424

*NETWORK_FAILURE_DETECTION*

When there is a primary and a standby network card, Serviceguard needs to determine when a card has failed, so it knows whether to fail traffic over to the other card. To detect failures, Serviceguard's Network Manager monitors both inbound and outbound traffic. The Manager will mark the card DOWN and begin to

attempt a failover when network traffic is not noticed for a time. (Serviceguard calculates the time depending on the type of LAN card.)

The configuration file specifies one of two ways to decide when the network interface card has failed:

- `INOUT` - The default method will count packets sent by polling, and declare a card down only the count stops incrementing for *both* the inbound *and* the outbound packets.
- `INONLY_OR_INOUT` - This option includes the INOUT method. However, it will also declare a card down if only the inbound count stops. With this method, Serviceguard tries to validate inbound failure reports by doing additional remote polling. (New in Serviceguard Version A.11.16.)

The default is `INOUT`.

The suitability of the new option depends mainly on your network configuration. See “Monitoring LAN Interfaces and Detecting Failure” on page 97 for more information. Also see “Inbound Failure Detection Enhancement” in the White Papers section of `http://docs.hp.com/hpux/ha` -> Serviceguard.

*`Kind of LAN Traffic`*

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

- Heartbeat
- Client Traffic
- Standby

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.

### RS232 Information

If you plan to configure a serial line (RS232), you need to determine the serial device file that corresponds with the serial port on each node.

1. If you are using a MUX panel, make a note of the system slot number that corresponds to the MUX and also note the port number that appears next to the selected port on the panel.
2. On each node, use `ioscan -fnC tty` to display hardware addresses and device file names. For example:

   ```
   # ioscan -fnC tty
   ```

   This lists all the device files associated with each RS232 device on a specific node.
3. Once you have identified the device files, verify your connection as follows. Assume that node 1 uses /dev/tty0p0, and node 2 uses /dev/tty0p6.
   - From a terminal on node 1, issue the following command:

     ```
     # cat < /dev/tty0p0
     ```
   - From a terminal on node 2, issue the following command:

     ```
     # cat /etc/passwd > /dev/tty0p6
     ```

     The contents of the password file should be displayed on the terminal on node 1.
4. On the worksheet, enter the following:

*RS232 Device File*

Enter the device file name corresponding to the serial interface to be used on this node. This parameter is known as `SERIAL_DEVICE_FILE` in the ASCII configuration file.

*Second Node Name*

Enter the name of the node that will be connected to this node via RS232. a node name can be 31 bytes or less.

## Setting SCSI Addresses for the Largest Expected Cluster Size

SCSI standards define priority according to SCSI address. To prevent controller starvation on the SPU, the SCSI interface cards must be configured at the highest priorities. Therefore, when configuring a highly available cluster, you should give nodes the highest priority SCSI addresses, and give disks addresses of lesser priority.

For SCSI, high priority starts at seven, goes down to zero, and then goes from 15 to eight. Therefore, seven is the highest priority and eight is the lowest priority. For example, if there will be a maximum of four nodes in the cluster, and all four systems will share a string of disks, then the SCSI address must be uniquely set on the interface cards in all four systems, and must be high priority addresses. So the addressing for the systems and disks would be as follows:

**Table 4-1 SCSI Addressing in Cluster Configuration**

| System or Disk | Host Interface SCSI Address |
|---|---|
| Primary System A | 7 |
| Primary System B | 6 |
| Primary System C | 5 |
| Primary System D | 4 |
| Disk #1 | 3 |
| Disk #2 | 2 |
| Disk #3 | 1 |
| Disk #4 | 0 |
| Disk #5 | 15 |
| Disk #6 | 14 |
| Others | 13 - 8 |

**NOTE** When a boot/root disk is configured with a low-priority address on a shared SCSI bus, a system panic can occur if there is a timeout on accessing the boot/root device. This can happen in a cluster when many nodes and many disks are configured on the same bus.

The correct approach is to assign SCSI addresses in such a way that the interface cards on cluster nodes have the highest priority SCSI addresses, followed by any boot/root disks that are on the shared bus, followed by all other disks on the shared bus.

## 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 Fibre Channel, single-ended SCSI, F/W SCSI, Ultra2 and Ultra160 SCSI.

*Slot Number* Indicate the slot number in which the interface card is inserted in the backplane of the computer.

*Address* Enter the bus hardware path number, which will be seen on the system later when you use `ioscan` to display hardware.

*Disk Device File* Enter the disk device file name. To display the name use the `ioscan -fnC disk` command.

Information from this section of the worksheet is used in creating the mirrored disk configuration using Logical Volume Manager. 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:

- `diskinfo`
- `ioscan -fnC disk`
- `lssf /dev/*dsk/c*`
- `bdf`
- `mount`
- `swapinfo`

- `vgdisplay -v`
- `lvdisplay -v`
- `lvlnboot -v`
- `vxdg list` (VxVM and CVM)
- `vxprint` (VxVM and CVM)

These are standard HP-UX commands. See their man pages for information of 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 storage group and cluster configuration. A printed listing of the output from the `lssf` command can be marked up to indicate which physical volume group a disk should be assigned to.

## Hardware Configuration Worksheet

The following worksheet 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.

```
    SPU Information:

    Host Name ____ftsys9___________ Series No ______rp8400____________

    Memory Capacity ____128 MB _________    Number of I/O Slots ______12_______

================================================================================
    LAN Information:

    Name of           Name of              Node IP              Traffic
    Subnet __Blue___  Interface ___lan0___ Addr___35.12.16.10__ Type   ____HB___

    Name of           Name of              Node IP              Traffic
    Subnet __Blue___  Interface ___lan2___ Addr_________________ Type   _standby_

    Name of           Name of              Node IP              Traffic
    Subnet __Red____  Interface ___lan1___ Addr___35.12.15.12_  Type _HB, client

================================================================================
    Network Failure Dections: ____INOUT___

================================================================================

    Serial (RS232) Heartbeat Interface Information:

    RS232 Device File ___/dev/tty0p0__
```

```
    Second Node Name ____ftsys10__________

================================================================================
    Disk I/O Information for Shared Disks:

    Bus Type _SCSI_ Slot Number _4__ Address _16_ Disk Device File __c0t1d0_

    Bus Type _SCSI_ Slot Number _6_  Address _24_ Disk Device File __c0t2d0_

    Bus Type ______ Slot Number ___  Address ____ Disk Device File  _________

    Attach a printout of the output from the ioscan -fnC disk command
    after installing disk hardware and rebooting the system.  Mark this
    printout to indicate which physical volume group each disk belongs to.
```

# 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 sources (UPS). Loss of a power circuit should not bring down the cluster.

Frequently, servers, mass storage devices, and other hardware have two or three separate power supplies, so they can survive the loss of power to one or more power supplies or power circuits. If your cluster has devices with redundant power supplies, connect the each devices power supplies to separate power circuits. This way the failure of a single power circuit will not cause the complete failure of any critical device in the cluster. For example, if all devices in a cluster have three power supplies, it will require a minimum of three separate power circuits to eliminate the power as a single point of failure for the cluster.

For services with only one power supply, 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 disk lock 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 cluster lock. Be sure that quorum server nodes have separate power sources from every cluster they serve. Be sure that quorum server or the cluster lock disks that will be configured as members of separate physical volume groups 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:

| | |
|---|---|
| *Host Name* | Enter the host name for each SPU. |
| *Disk Unit* | Enter the disk drive unit number for each disk. |
| *Tape Unit* | Enter the tape unit number for each backup device. |

*Other Unit* Enter the number of any other unit.

*Power Supply* 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 following worksheet 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.

```
=============================================================================
SPU Power:

Host Name ____ftsys9__________    Power Supply _____1________________

Host Name ____ftsys10_________    Power Supply _____2________________

=============================================================================
Disk Power:

Disk Unit __________1________________    Power Supply ____3________________

Disk Unit __________2________________    Power Supply ____4________________

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 Planning

The quorum server (QS) provides tie-breaking services for clusters. The QS is described in chapter 3 under “Use of the Quorum Server as the Cluster Lock.”

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.

**NOTE**

It is recommended that the node on which the quorum server 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 are not likely to fail at the same time.

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. The quorum server has to be configured on an IPv4 network. IPv6 addresses are not supported.

*`Supported Node Names`*

Enter the names (31 bytes or less) 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 following worksheet 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.

```
     Quorum Server Data:
==============================================================================

    QS Hostname: _________________IP Address: _____________________


==============================================================================

    Quorum Services are Provided for:


Cluster Name: ___________________________________________________________

    Host Names ____________________________________________

    Host Names ____________________________________________


Cluster Name: ___________________________________________________________

    Host Names ____________________________________________

    Host Names ____________________________________________
```

# LVM Planning

You can create storage groups using the HP-UX Logical Volume Manager (LVM), or using VERITAS VxVM and CVM software, which are described in the next section.

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

- The root disk should belong to its own volume group.
- The volume groups that contain high availability applications, services, or data must be on a bus or busses available to the primary node and all adoptive nodes.
- High availability applications, services, and data should be placed in a separate volume groups from non-high availability applications, services, and data.
- You must group high availability applications, services, and data, whose control needs to be transferred together, onto 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, onto the same volume group.
- Your root disk must not belong to a volume group that can be activated on another node.
- It is recommended 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 that they are associated with (for example, `/dev/vegetables` will simplify cluster administration.

If you plan to use the EMS HA Disk Monitor, refer to the section on "Rules for Using EMS Disk Monitor with Serviceguard" in the manual *Using High Availability Monitors* (B5736-90025).

## LVM Worksheet

The following worksheet will help you organize and record your specific physical disk configuration. Make as many copies as you need. Fill out the worksheet and keep it for future reference. This worksheet only includes volume groups and physical volumes. The Package Configuration worksheet (presented later in this chapter) contains more space for recording information about the logical volumes and file systems that are part of each volume group.

```
=============================================================================

    Volume Group Name:      __________/dev/vg01_____________________________

    Name of First Physical Volume Group:  _______bus0_____________________

    Physical Volume Name:
_____________/dev/dsk/c1t2d0_____________________

    Physical Volume Name:
_____________/dev/dsk/c2t2d0_____________________

    Physical Volume Name:
_____________/dev/dsk/c3t2d0_____________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Name of Second Physical Volume Group: _______bus1_____________________

    Physical Volume Name:
_____________/dev/dsk/c4t2d0_____________________

    Physical Volume Name:
_____________/dev/dsk/c5t2d0_____________________

    Physical Volume Name:
```

```
_____________/dev/dsk/c6t2d0_____________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________

    Physical Volume Name:
_____________________________________________________
```

# CVM and VxVM Planning

You can create storage groups using the HP-UX Logical Volume Manager (LVM, described in the previous section), or using VERITAS VxVM and CVM software.

When designing a storage configuration using CVM or VxVM disk groups, consider the following:

- You must create a *rootdg* disk group on each cluster node that will be using VxVM storage. *This is not the same as the HP-UX root disk, if an LVM volume group is used*. The VxVM root disk group can only be imported on the node where it is created. This disk group is created only *once* on each cluster node.
- CVM disk groups are created after the cluster is configured, whereas VxVM disk groups may be created before cluster configuration if desired.
- High availability applications, services, and data should be placed in separate disk groups from non-high availability applications, services, and data.
- You must not group two different high availability applications, services, or data, whose control needs to be transferred independently, onto the same disk group.
- Your HP-UX root disk can belong to an LVM or VxVM volume group (starting from VxVM 3.5) that is *not* shared among cluster nodes.
- The cluster lock disk can only be configured with an LVM volume group.
- VxVM disk group names should not be entered into the cluster configuration ASCII file. These names are not inserted into the cluster configuration ASCII file by `cmquerycl`.

## CVM and VxVM Worksheet

The following worksheet will help you organize and record your specific physical disk configuration. Make as many copies as you need. Fill out the worksheet and keep it for future reference. This worksheet only

includes volume groups and physical volumes. The Package Configuration worksheet (presented later in this chapter) contains more space for recording information about the logical volumes and file systems that are part of each volume group.

```
=============================================================================

   Disk Group Name:      _______dg01_____________________________________

   Disk Name:   _____________________c1t2d0_____________________________

   Disk Name:   _____________________c2t2d0_____________________________

   Disk Name:   _____________________c3t2d0_____________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________

   Disk Group Name:      _______dg02_____________________________________

   Disk Name:   _____________________c1t3d0_____________________________

   Disk Name:   _____________________c2t3d0_____________________________

   Disk Name:   _____________________c3t3d0_____________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________

   Disk Name:   _____________________________________________________
```

# 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. 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 availability of raw disk access. Applications that use raw disk access should be designed with crash recovery services.
- 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` files, must be the same on all nodes.

The Serviceguard Extension for Faster Failover is a purchased product that can optimize failover time for certain two-node clusters. The clusters must be configured to meet certain requirements. When installed, the product is enabled by a parameter in the cluster configuration file. Release Notes for the product are posted at `http://docs.hp.com -> high availability`.

## Heartbeat Subnet and Re-formation Time

The speed of cluster re-formation is partially dependent on the type of heartbeat network that is used. Ethernet results in a slower failover time than the other types. If two or more heartbeat subnets are used, the one with the fastest failover time is used.

## Cluster Lock Information

The purpose of the cluster lock is to ensure that only one new cluster is formed in the event that exactly half of the previously clustered nodes try to form a new cluster. It is critical that only one new cluster is formed and that it alone has access to the disks specified in its packages. You can specify either a lock disk or a quorum server as the cluster lock.

A one-node cluster does not require a lock. Two-node clusters require the use of a cluster lock, but the lock is recommended for larger clusters as well. Clusters larger than 4 nodes can only use a quorum server as the cluster lock.

### Cluster Lock Disk and Re-formation Time

If you are using a lock disk, the acquisition of the cluster lock disk takes different amounts of time depending on the disk I/O interface that is used. After all the disk hardware is configured, but before configuring the cluster, you can use the `cmquerycl` command specifying all the nodes in the cluster to display a list of available disks and the re-formation time associated with each. Example:

```
# cmquerycl -v -n ftsys9 -n ftsys10
```

Alternatively, you can use SAM to display a list of cluster lock physical volumes, including the re-formation time.

By default, Serviceguard selects the disk with the fastest re-formation time. But you may need to choose a different disk because of power considerations. Remember that the cluster lock disk should be separately powered, if possible.

### Cluster Lock Disks and Planning for Expansion

You can add additional cluster nodes after the cluster is up and running, but doing so without bringing down the cluster requires you to follow some rules. Recall that a cluster with more than 4 nodes may not have a lock disk. Thus, if you plan to add enough nodes to bring the total to more than 4, you should use a quorum server.

## 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 can be 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 parameter names given below are the names that appear in the cluster ASCII configuration file.

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 characters.

*QS_HOST*

The name or IP address of a host system outside the current cluster that is providing quorum server functionality. This parameter is only used when you employ a quorum server for tie-breaking services in the cluster.

*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).

*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.

*FIRST_CLUSTER_LOCK_VG, SECOND_CLUSTER_LOCK_VG*

The volume group containing the physical disk volume on which a cluster lock is written. Identifying a cluster lock volume group is essential in a two-node cluster. If you are creating two cluster locks, enter the volume group name or names for both locks. This parameter is only used when you employ a lock disk for tie-breaking services in the cluster.

Use FIRST_CLUSTER_LOCK_VG for the first lock volume group. If there is a second lock volume group, the parameter SECOND_CLUSTER_LOCK_VG is included in the file on a separate line.

**NOTE**

Lock volume groups must also be defined in VOLUME_GROUP parameters in the cluster ASCII configuration file.

*NODE_NAME*

The hostname of each system that will be a node in the cluster. The node name can be up to 31 bytes long. The node name must not contain the full domain name. For example, enter ftsys9, not ftsys9.cup.hp.com.

*NETWORK_INTERFACE*

The name of each LAN that will be used for heartbeats or for user data. An example is lan0.

*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. A heartbeat IP address can only be an IPv4 address.

If you will be using VERITAS CVM disk groups for storage, you can only use a *single* heartbeat subnet. In this case, the heartbeat should be configured with standby LANs or as a group of aggregated ports.

**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 HP-UX. 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.

A stationary IP address can be either an IPv4 or an IPv6 address. For more details of IPv6 address format, see the Appendix , “IPv6 Address Types,” on page 424

*FIRST_CLUSTER_LOCK_PV, SECOND_CLUSTER_LOCK_PV*

The name of the physical volume within the Lock Volume Group that will have the cluster lock written on it. This parameter is `FIRST_CLUSTER_LOCK_PV` for

the first physical lock volume and SECOND_CLUSTER_LOCK_PV for the second physical lock volume. If there is a second physical lock volume, the parameter SECOND_CLUSTER_LOCK_PV is included in the file on a separate line. These parameters are only used when you employ a lock disk for tie-breaking services in the cluster.

Enter the physical volume name as it appears on both nodes in the cluster (the same physical volume may have a different name on each node). If you are creating two cluster locks, enter the physical volume names for both locks. The physical volume group identifier can contain up to 39 characters.

*SERIAL_DEVICE_FILE*

The name of the device file that corresponds to serial (RS232) port that you have chosen on each node. Specify this parameter when you are using RS232 as a heartbeat line.

In the ASCII cluster configuration file, this parameter is SERIAL_DEVICE_FILE. The device file name can contain up to 39 characters.

*HEARTBEAT_INTERVAL*

The normal interval between the transmission of heartbeat messages from one node to the other in the cluster. Enter a number of seconds.

In the ASCII cluster configuration file, this parameter is HEARTBEAT_INTERVAL, and its value is entered in microseconds.

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. This parameter is entered in microseconds.

Default value is 2,000,000 microseconds in the ASCII file. Minimum is 2 * (Heartbeat Interval). The maximum recommended value for this parameter is 30,000,000 in the ASCII file, or 30 seconds in Serviceguard Manager. 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. In the ASCII cluster configuration file, this parameter is AUTO_START_TIMEOUT. 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 in the ASCII file (600 seconds in Serviceguard Manager).

*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 (2 seconds in Serviceguard Manager). 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.

*MAX_CONFIGURED_PACKAGES*

This parameter sets the maximum number of packages that can be configured in the cluster. In the ASCII cluster configuration file, this parameter is known as MAX_CONFIGURED_PACKAGES.

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 100 KB of lockable memory on all cluster nodes. Be sure to add one for the VxVM-CVM-pkg if you are using CVM disk storage.

**NOTE**

Remember to tune HP-UX kernel parameters on each node to ensure that they are set high enough for the largest number of packages that would ever run concurrently on that node.

*VOLUME_GROUP*

The name of an LVM volume group whose disks are attached to at least two nodes in the cluster. Such disks are considered cluster aware. In the ASCII cluster configuration file, this parameter is VOLUME_GROUP. The volume group name can have up to 39 characters.

*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 “Editing Security Files” on page 182.

*FAILOVER_OPTIMIZATION*

You will only see this parameter if you have installed Serviceguard Extension for Faster Failover, a separately purchased product. You enable the product by setting this parameter to TWO_NODE. Default is disabled, set to NONE. For more information about the product and its cluster configuration requirements, go to http://www.docs.hp.com/hpux/ha and click Serviceguard Extension for Faster Failover.

*NETWORK_FAILURE_DETECTION*

When there is a primary and a standby network card, Serviceguard needs to determine when a card has failed, so it knows whether to fail traffic over to the other card. To detect failures, Serviceguard’s Network Manager monitors both inbound and outbound traffic. The Manager will mark the card DOWN and begin to attempt a failover when network traffic is not noticed for a time. (Serviceguard calculates the time depending on the type of LAN card.)

The configuration file specifies one of two ways to decide when the network interface card has failed:

- `INOUT` - The default method will count inbound and outbound failures separately, and declare a card down only when both have reached a critical level.
- `INONLY_OR_INOUT` - This option combines the inbound and outbound failure counts, and will declare a card down when the total failures reach a critical amount, regardless of their source. With this method, Serviceguard tries to validate inbound failure reports by doing additional remote polling.

The default is `INOUT`.

The suitability of an option depends mainly on your network configuration. To see more about whether the new INONLY_OR _INOUT option is the best for your network configuration, see “Inbound Failure Detection Enhancement” `http://docs.hp.com/hpux/ha` -> Serviceguard White Papers.

## Cluster Configuration Worksheet

The following worksheet will help you to organize and record your cluster configuration.

```
 Name and Nodes:
================================================================================
    Cluster Name: ___ourcluster_______________

    Node Names: ____node1___________________    ____node2___________________

    Maximum Configured Packages: ______12________
================================================================================
     Quorum Server Data:
================================================================================
    Quorum Server Host Name or IP Address: __lp_qs ____________________

    Quorum Server Polling Interval: _300000000_ microseconds

    Quorum Server Timeout Extension: _______________ microseconds
================================================================================
    Subnets:
================================================================================
    Heartbeat Subnet: ___15.13.168.0______

    Monitored Non-heartbeat Subnet: _____15.12.172.0___

    Monitored Non-heartbeat Subnet: ____________________
================================================================================
    Cluster Lock Volume Groups and Volumes:
================================================================================
    First Lock Volume Group:     |        Physical Volume: |
        _________________        |   Name on Node 1: ____________________
                                 |
                                 |   Name on Node 2: ____________________
                                 |
                                 |   Disk Unit No: _________
                                 |
```

```
                                |   Power Supply No: ________
=============================================================================
    Timing Parameters:
=============================================================================
    Heartbeat Interval: _1 sec_
=============================================================================
    Node Timeout: _2 sec_
=============================================================================
    Network Polling Interval: _2 sec_
    Metwork Monitor  _INOUT_
=============================================================================
    Autostart Delay: _10 min___
=============================================================================
    Cluster Aware LVM Volume Groups _____________________________________________
=============================================================================
   Access Policies
     User: __ ANY_USER
     Host: __ ftsys9__
     Role: __ full_admin__

     User: __ sara itgrp lee __
     Host: __ ftsys10__
     Role: __ package_admin__
=============================================================================
```

# 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** LVM 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." VERITAS disk groups that are to be activated by packages must be defined in the package configuration ASCII file, described below.

## 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.

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 filesystems at different times.

It is recommended that you use customized logical volume names that are different from the default logical volume names (lvol1, lvol2, etc.). Choosing logical volume names that represent the high availability applications that they are associated with (for example, `lvoldatabase`) will simplify cluster administration.

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:

```
# /dev/vg01/lvoldb1 /applic1 vxfs defaults 0 1   # These six e
ntries are
# /dev/vg01/lvoldb2 /applic2 vxfs defaults 0 1   # for informa
tion purposes
# /dev/vg01/lvoldb3 raw_tables ignore ignore 0 0 # only. They
record the
# /dev/vg01/lvoldb4 /general vxfs defaults 0 2   # logical vol
umes that
# /dev/vg01/lvoldb5 raw_free ignore ignore 0 0   # exist for S
erviceguard's
# /dev/vg01/lvoldb6 raw_free ignore ignore 0 0   # HA package.
 Do not uncomment.
```

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.”

## Parameters for Configuring EMS Resources

Serviceguard provides a set of parameters for configuring EMS resources. These are RESOURCE_NAME, RESOURCE_POLLING_INTERVAL, RESOURCE_START, and RESOURCE_UP_VALUE. Enter these parameters to the package configuration file for each resource the package will be dependent on. The DEFERRED_RESOURCE_NAME is added to the package control script for any resource that has a RESOURCE_START setting of DEFERRED.

The RESOURCE_START option is used to determine when Serviceguard should start up resource monitoring for EMS resources. The RESOURCE_START option can be set to either AUTOMATIC or DEFERRED.

If AUTOMATIC is specified, Serviceguard will start up resource monitoring for these resources automatically when the Serviceguard cluster daemon starts up on the node. If resources are configured AUTOMATIC, you do not need to define DEFERRED_RESOURCE_NAME in the package control script.

If DEFERRED is selected, Serviceguard will not attempt to start resource monitoring for these DEFERRED resources during node start up. However, these DEFERRED resources must be specified in the package control script, setting the DEFERRED_RESOURCE_NAME parameter, so that the DEFERRED resources will be started up from the package control script during the package run time.

The following is an example of how to configure DEFERRED and AUTOMATIC resources. In the package configuration file, specify resources as follows:

```
RESOURCE_NAME               /net/interfaces/lan/status/lan0
RESOURCE_POLLING_INTERVAL   60
RESOURCE_START              DEFERRED
RESOURCE_UP_VALUE           = UP


RESOURCE_NAME               /net/interfaces/lan/status/lan1
RESOURCE_POLLING_INTERVAL   60
RESOURCE_START              DEFERRED
RESOURCE_UP_VALUE           = UP


RESOURCE_NAME               /net/interfaces/lan/status/lan2
RESOURCE_POLLING_INTERVAL   60
```

```
RESOURCE_START              AUTOMATIC
RESOURCE_UP_VALUE           = UP
```

In the package control script, specify only the deferred resources, using the DEFERRED_RESOURCE_NAME parameter:

```
DEFERRED_RESOURCE_NAME[0]="/net/interfaces/lan/status/lan0"
DEFERRED_RESOURCE_NAME[1]="/net/interfaces/lan/status/lan1"
```

## 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 100KB of lockable memory on each cluster node.

## Choosing Switching and Failover Behavior

Switching IP addresses from a failed LAN card to a standby LAN card on the same physical subnet may take place if Automatic Switching is set to Enabled in Serviceguard Manager (LOCAL_LAN_FAILOVER_ALLOWED set to YES in the ASCII package configuration file). Automatic Switching Enabled is the default.

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.

Table 3-3 in the previous chapter describes different types of failover behavior and the settings in Serviceguard Manager or 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 parameter names given below are the names that appear in Serviceguard Manager. The names coded in

the ASCII cluster configuration file appear at the end of each entry. 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. In the ASCII package configuration file, this parameter is PACKAGE_NAME.

In the package ASCII file, the package name can be from 1 to 39 characters. It must not contain any of the following illegal characters: space, slash (/), backslash(\), and asterisk (*). All other characters are legal.

In Serviceguard Manager, more non-alphanumeric characters are not allowed. See the online help topic "Configuring Packages."

*PACKAGE_TYPE*

The type of the package. This parameter indicates whether the package will run on one node at a time or on multiple nodes. Valid types are FAILOVER and SYSTEM_MULTI_NODE. Default is FAILOVER.

The SYSTEM_MULTI_NODE type is used only to define the VxVM-CVM-pkg package that supports shared CVM data storage among all cluster nodes. You cannot create a user-defined package with a type of SYSTEM_MULTI_NODE.

*Failover policy*

The policy used by the package manager to select the node on which the package should be automatically started. In the ASCII package configuration file, this parameter is known as FAILOVER_POLICY.

Default is CONFIGURED_NODE, which selects the next available node in the list of node names for the package. The order of preference is the order of the node name entries in the package's configuration file.

The alternate policy is MIN_PACKAGE_NODE, which creates a list of packages running on each node that can run this package, and selects the node that is running the fewest packages.

*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. In the ASCII package configuration file, this parameter is known as FAILBACK_POLICY.

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. (This is the same behavior as in previous versions of Serviceguard.) 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.

*Package nodes*

The names of primary and alternate nodes for the package. In the ASCII configuration file, this parameter is NODE_NAME. 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.

If all nodes in the cluster are to be used, and if order is not important, you can specify: NODE_NAME *

A node name can be up to 31 bytes long.

*Package auto run*

In the ASCII package configuration file, the AUTO_RUN parameter was formerly known as PKG_SWITCHING_ENABLED. The parameter determines how a package may start up. If Automatic Switching is enabled (AUTO_RUN 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 Automatic Switching is set to Disabled (AUTO_RUN 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 AUTO_RUN YES in the ASCII file, which allows a package to start up normally on an available node. Check the box to enable auto-run in Serviceguard Manager; enter AUTO_RUN YES or NO in the ASCII file.

*Local LAN failover*

Enter Enabled or Disabled. In the event of a failure, this permits Serviceguard to switch LANs locally, that is, transfer the package IP address to a standby LAN card. The default is Enabled. In the ASCII package configuration file, this parameter is called LOCAL_LAN_FAILOVER_ALLOWED, and possible values are YES and NO.

Default is checked in Serviceguard Manger, or YES in the ASCII file.

*Node fail fast*

In the event of the failure of the control script itself, or the failure of a subnet, or the report of an EMS monitor event showing that a resource is down, if this parameter is set to Enabled, Serviceguard will issue a TOC on the node where the control script fails. In the ASCII package configuration file, this parameter is called NODE_FAIL_FAST_ENABLED, and possible values are YES and NO.

The default is Disabled, not checked in Serviceguard Manger, or NO in the ASCII file.

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
- An EMS resource fails
- The halt script does not exist
- Serviceguard is unable to execute the halt script
- The halt script or the run 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.

*Controlscript pathname*

Serviceguard Manager will automatically create and maintain a control script if you use Guided Mode, the Serviceguard Manager default. If you choose, you can edit the control script yourself, and name your own path. Once you leave Guided Mode, however, Serviceguard Manager cannot update your script.

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. In the ASCII package configuration file, this parameter maps to the two separate parameters named RUN_SCRIPT and HALT_SCRIPT.

It is recommended that you use the same script as both the run and halt script. This script will contain both your package run instructions and your package halt instructions. If you wish to separate the package run instructions and package halt instructions into separate scripts, the package configuration file allows allows you to do this by naming two separate scripts. However, under most conditions, it is simpler to use the name of the single control script as the name of the

RUN_SCRIPT and the HALT_SCRIPT in the ASCII file. 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'.

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. In the ASCII configuration file, these parameters are RUN_SCRIPT_TIMEOUT and HALT_SCRIPT_TIMEOUT. 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 HP-UX 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.

**NOTE** VxVM disk groups are imported at package run time and exported at package halt time. If you are using a large number of VxVM disks in your package, the timeout must be high enough to allow all of them to finish the import or export.

*CVM diskgroups*

This parameter is used for CVM disk groups. Enter the names of all the CVM disk groups the package will use.

In the ASCII package configuration file, this parameter is called STORAGE_GROUP.

**NOTE** You should not enter the names of LVM volume groups or VxVM disk groups in the package ASCII configuration file.

*Service name*

Enter a unique name for each service. In the ASCII package configuration file, this parameter is called SERVICE_NAME.

Define one SERVICE_NAME entry for each service.You can configure a maximum of 30 services per package and 900 services per cluster. 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 characters.

*Service fail fast*

Enter Enabled or Disabled 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 Disabled.

In the ASCII package configuration file, this parameter is SERVICE_FAIL_FAST_ENABLED, and possible values are YES and NO. 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. In the ASCII package configuration file, this parameter is SERVICE_HALT_TIMEOUT.

If you do not fill in a number, Serviceguard will not allow any timeout (0 seconds). The maximum value is restricted only by the HP-UX 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.

In the ASCII package configuration file, this parameter is called SUBNET.

*EMS resource*

The name of a resource that is to be monitored by Serviceguard as a package dependency. In the ASCII package configuration file, this parameter is called RESOURCE_NAME.

A resource name is the name of an important attribute of a particular system resource. The resource name includes the entire hierarchy of resource class and subclass within which the resource exists on a system. Obtain the resource name from the list provided in

Serviceguard Manager in the EMS tab's Browse button ("Available EMS resources"), or obtain it from the documentation supplied with the resource monitor.

A maximum of 60 resources may be defined per *cluster*. Note also the limit on Resource Up Values described below.

Maximum length of the resource name string is 1024 characters.

*Resource polling interval*

The frequency of monitoring a configured package resource. In the ASCII package configuration file, this parameter is called RESOURCE_POLLING_INTERVAL.

Default is 60 seconds. The minimum value is 1. The maximum value has no functional limit.

The Resource Polling Interval appears on the list provided in Serviceguard Manger's package configuration screen, EMS Resource tab's Browse button ("Description of selected EMS resource"), or you can obtain it from the documentation supplied with the resource monitor.

*Resource start*

An attribute of a resource that determines whether it should start up before or after the package starts. In the ASCII package configuration file, this parameter is called RESOURCE_START.

Default setting is AUTOMATIC, which means that the resource starts at the time the node joins the cluster. The other possible setting is DEFERRED, which means that the package services will start up *before* the resource starts. If a resource is configured with DEFERRED startup, the name of the resource has to be added to the control script's DEFERRED_RESOURCE_NAME parameter. (If you use Serviceguard Manger's guided mode to create the control script automatically, it will be entered for you.)

*Resource UP criteria*

The criteria for judging whether an additional package resource has failed or not. In the ASCII package configuration file, this parameter is called RESOURCE_UP_VALUE. The Resource Up Value appears on the "Description of selected EMS resources" list provided in Serviceguard Manger's EMS Browse button, or you can obtain it from the documentation supplied with the resource monitor.

You can configure a total of 15 Resource Up Values per package. For example, if there is only one resource in the package, then a maximum of 15 Resource Up Values can be defined. If there are two Resource Names defined and one of them has 10 Resource Up Values, then the other Resource Name can have only 5 Resource Up Values.

The maximum length of the Resource Up Value string is 1024 characters.

*Access control policies*

You can configure package administration access for this package. Be sure the policy is not conflicting or redundant to an access policy defined in the cluster configuration file. For more information, see "Editing Security Files" on page 182.

### Package Configuration Worksheet

Assemble your package configuration data in a separate worksheet for each package, as shown in the following example.

```
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: __/etc/cmcluster/pkg1/control.sh__Timeout: _NO_TIMEOUT_

Package Halt Script: __/etc/cmcluster/pkg1/control.sh_Timeout: _NO_TIMEOUT_

Package AutoRun Enabled?  __YES___ Local LAN Failover Allowed?  ___YES__

Node Failfast Enabled?      ____NO____

CVM Storage Groups:

_____________________________________________________________________

Additional Package Resource:

Resource Name:___________ Polling Interval_______ Resource UP Value_________
_____________________________________________________________________
Access Policies:
User:___any_user____ From node:___ftsys9____ Role:__package_admin____
User:___lee ron admn__ From node:__ftsys10__ Role:__package_admin____
_____________________________________________________________________
```

## Package Control Script Variables

The control script that accompanies each package must also be edited to assign values to a set of variables. The following variables can be set:

*PATH*

Specify the path to be used by the script.

VGCHANGE

Specifies the method of activation for LVM volume groups. Leave the default (VGCHANGE="vgchange -a e") if you want volume groups activated in exclusive mode. This assumes the volume groups have been initialized with vgchange -c y at the time of creation.

Use VGCHANGE="vgchange -a e -q n" if your disks are mirrored on separate physical paths.

Use VGCHANGE="vgchange -a e -q n -s" if your disks are mirrored on separate physical paths and you want the mirror resynchronization to occur in parallel with the package startup.

Use VGCHANGE="vgchange -a y" if you wish to use non-exclusive activation mode. Single node cluster configurations must use non-exclusive activation.

*CVM_ACTIVATION_CMD*

Specifies the command for activation of VERITAS CVM disk groups.

Use the default CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=exclusivewrite" if you want disk groups activated in exclusive write mode.

Use CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=readonly" if you want disk groups activated in read-only mode.

Use CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedread" if you want disk groups activated in shared read mode.

Use CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedwrite" if you want disk groups activated in shared write mode.

**NOTE** VxVM disk groups do not allow you to select specific activation commands. The VxVM disk group activation always uses the same command.

*VXVOL*

Controls the method of mirror recovery for mirrored VxVM volumes.

Use the default VXVOL="vxvol -g \$DiskGroup startall" if you want the package control script to wait until recovery has been completed.

Use VXVOL="vxvol -g \$DiskGroup -o bg startall" if you want the mirror resynchronization to occur in parallel with the package startup.

*Volume Groups*

This array parameter contains a list of the LVM volume groups that will be activated by the package. Enter each VG on a separate line.

*CVM Disk Groups*

This array parameter contains a list of the VERITAS CVM disk groups that will be used by the package. Enter each disk group on a separate line.

*VxVM Disk Groups*

This array parameter contains a list of the VERITAS VxVM disk groups that will be activated by the package. Enter each disk group on a separate line.

*Logical Volumes, File Systems and Mount Options*

These array parameters, entered together as triplets into array variables. Each triplet specifies a filesystem, a logical volume, and a mount option string for a file system used by the package. In the package control script file, these variables are arrays, as follows: `LV`, `FS` and `FS_MOUNT_OPT`.

On starting the package, the script may activate one or more storage groups, and it may mount logical volumes onto file systems. At halt time, the script unmounts the file systems and deactivates each storage group. All storage groups must be accessible on each target node (CVM disk groups must be accessible on all nodes in the cluster). For each file system (`FS`), you must identify a logical volume (`LV`).

A logical volume can be built on an LVM volume group, a VERITAS CVM disk group, or a VERITAS VxVM disk group. LVs can be entered in any order, regardless of the type of storage group that is used.

*Filesystem Unmount Count*

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

*Filesystem Mount Retry Count*

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 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_VGCHANGE_OPERATIONS

Specifies the number of concurrent volume group activations or deactivations to allow during package startup or shutdown. The default is 1. Setting this variable to a higher value may improve performance if you are using a large number of volume groups. 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_DISKGROUP_OPERATIONS

Specifies the number of concurrent VxVM disk group imports or deports to allow during package startup or shutdown. The default is 1. Setting this variable to a higher value may improve performance if you are using a large number of disk groups. 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_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_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. Package IP addresses may be either IPv4 addresses or IPv6 addresses. For more details of IPv6 address format, see the Appendix , "IPv6 Address Types," on page 424

In the package control script file, these variables are entered in pairs. An IPv4 example is `IP[0]=192.10.25.12` and `SUBNET[0]=192.10.25.0`. (In this case the subnet mask is 255.255.255.0.) An IPv6 examples is `IP[0]=2001::3` and `SUBNET[0]=2001::/64`. (In this case the subnet mask is ffff:ffff:ffff:ffff::.)

*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 characters.

*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 "-R " indicates an infinite number of retries. 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.

*Deferred Resource Names*

For each deferred resource specified in the package configuration ASCII file, you must enter the resource name in this array in the control script. The name should be spelled exactly as it is spelled in the `RESOURCE_NAME` parameter in the package ASCII configuration file.

In the package control script file, enter the value into the array known as `DEFERRED_RESOURCE_NAME`. This name must match the resource name listed with the `RESOURCE_START` parameter in the package ASCII configuration file.

DTC Manager Data

Enter a DTC Name for each DTC. For information on using a DTC with Serviceguard, see the chapter entitled “Configuring DTC Manager for Operation with Serviceguard” in the manual *Using the HP DTC Manager/UX*.

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.

### Control Script Worksheet

Assemble your package control script data in a separate worksheet for each package, as in the following example.

```
    LVM Volume Groups:

    VG[0]_______________VG[1]_______________VG[2]_______________

    VGCHANGE: _____________________________________________

    CVM Disk Groups:

    CVM_DG[0]___/dev/vx/dg01____CVM_DG[1]_____________CVM_DG[2]_______________

    CVM_ACTIVATION_CMD: ____________________________________________

    VxVM Disk Groups:

    VXVM_DG[0]___/dev/vx/dg01____VXVM_DG[1]____________VXVM_DG[2]_____________

================================================================================
    Logical Volumes and File Systems:
    LV[0]___/dev/vg01/lv011____FS[0]____/mnt1___________FS_MOUNT_OPT[0]__________

    LV[1]_____________________FS[1]_____________________FS_MOUNT_OPT[1]__________

    LV[2]_____________________FS[2]_____________________FS_MOUNT_OPT[2]__________

    FS Umount Count: _____________FS Mount Retry Count:_____________________

    CONCURRENT VCHANGE OPERATIONS: _____________________________________________-

    CONCURRENT DISKGROUP OPERATIONS: ___________________________________________-
```

```
    CONCURRENT MOUNT/UMOUNT OPERATIONS: _____________________________________

    CONCURRENT FSCK OPERATIONS: _____________________________________________

================================================================================
   Network Information:
    IP[0] ____15.13.171.14____ SUBNET ____15.13.168___________

    IP[1] _____________________ SUBNET _____________________
================================================================================
    Service Name: __svc1____  Command: ___/usr/bin/MySvc -f__ Restart: _-r 2___

    Service Name: _______ Command: _________ Restart: __

    Deferred Resource Name ___________________
```

# 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:

- Preparing Your Systems
- Setting up the Quorum Server
- Installing Serviceguard
- Creating a Storage Infrastructure with LVM
- Creating a Storage Infrastructure with VxVM
- Configuring the Cluster
- Creating a Storage Infrastructure with CVM
- Managing the Running Cluster

Configuring packages is described in the next chapter.

If you use Serviceguard Manager, the graphical user interface, you can configure clusters with Serviceguard version 11.16 or later. To create a cluster, connect to a session server with Serviceguard A.11.16 or later. One the map or tree, select a cluster or unused node with Serviceguard A.11.16 or later. Go to the Actions menu. You will see a screen with several tabs. The tabs prompt you for information, and show you discovered options for many of the choices. There is online Help available to help you make decisions.

If you are using Serviceguard commands to configure the cluster and packages, use the 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 Where Files Are Located

Serviceguard uses a special file, `/etc/cmcluster.conf`, to define the locations for configuration and log files within the HP-UX filesystem. The following locations are defined in the file:

```
###################### cmcluster.conf#########################
#
# Highly Available Cluster file locations
#
# This file must not be edited
################################################################

SGCONF=/etc/cmcluster
SGSBIN=/usr/sbin
SGLBIN=/usr/lbin
SGLIB=/usr/lib
SGCMOM=/opt/cmom
SGRUN=/var/adm/cmcluster
SGAUTOSTART=/etc/rc.config.d/cmcluster
SGCMOMLOG=/var/adm/syslog/cmom
```

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

Throughout this book, system filenames are usually given with one of these location prefixes. Thus, references to `$SGCONF/<FileName>` can be resolved by supplying the definition of the prefix that is found in this file. For example, if `SGCONF` is defined as `/etc/cmcluster/conf`, then the complete pathname for file `$SGCONF/cmclconfig` would be `/etc/cmcluster/conf/cmclconfig`.

**NOTE** Do *not* edit the `/etc/cmcluster.conf configuration` file.

## 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 systems 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.comgryf

10.8.1.131 gryf.uksr.hp.comgryf

15.145.162.132 sly.uksr.hp.comsly

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** If you use of fully qualified domain name (FQDN), Serviceguard will only recognize the hostname portion. 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 whitepaper “Securing Serviceguard” for more information.

To disable the use of `identd`, add the `-i` option to the `tcp hacl-cfg` and `hacl-probe inetd` configurations.

For example, on HP-UX with Serviceguard A.11.16

1. Change the `cmclconfd` entry in `/etc/inetd.conf` to appear as:
   `hacl-cfg stream tcp nowait root /usr/lbin/cmclconfd \ cmclconfd -c -i`.
2. Change the `cmomd` entry in `/etc/inetd.conf` to appear as:
   `hacl-probe stream tcp nowait root \ /opt/cmom/lbin/cmomd /opt/cmom/lbin/cmomd -i -f \ /var/opt/cmom/cmomd.log -r /var/opt/cmom`.
3. Restart `inetd`: `/etc/init.d/inetd 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 privledges 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 privledges 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 privledges of the Package Admin role.

If you upgrade a cluster to Version 11.16, the `cmclnodelist` entries are automatically updated into Access Control Policies in the cluster configuration file. All non-root user-hostname pairs will be given the role of Monitor (view only).

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 wish 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:

**Table 5-1 `cmclnodelist` 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.

## Defining Name Resolution Services

It is important to understand how Serviceguard uses name resolution services. When you employ any user-level Serviceguard command (including `cmviewcl`), the command uses name lookup to obtain the addresses of all the cluster nodes. If name services are not available, the command could hang or return an unexpected networking error message. In Serviceguard Manager, cluster or package operations also will return an error if name services are not available.

**NOTE** If such a hang or error occurs, Serviceguard and all protected applications will continue working even though the command you issued does not. That is, only the Serviceguard configuration commands and Serviceguard Manager functions are impacted, not the cluster daemon or package services.

To avoid this problem, you can use the `/etc/hosts` file on all cluster nodes in addition to DNS or NIS. It is also recommended to make DNS highly available either by using multiple DNS servers or by configuring DNS into a Serviceguard package.

To do this, add one of the following lines in the /etc/nsswitch.conf file:

- for DNS, enter (one line):

  ```
  hosts: dns [NOTFOUND=continue UNAVAIL=contine] dns
  {NOTFOUND=return UNAVAIL=return]
  ```

- for NIS, enter (one line):

  ```
  hosts: nis [NOTFOUND=continue UNAVAIL=contine] nis
  {NOTFOUND=return UNAVAIL=return]
  ```

A workaround for the problem that still retains the ability to use conventional name lookup is to configure the `/etc/nsswitch.conf` file to search the `/etc/hosts` file when other lookup strategies are not working. In case name services are not available, Serviceguard commands will then use the `/etc/hosts` file on the local system to do name resolution. Of course, the names and IP addresses of all the nodes in the cluster must be in the `/etc/hosts` file.

### Name Resolution Following Primary LAN Failure or Loss of DNS

There are some special configuration steps required to allow cluster configuration commands such as `cmrunnode` and `cmruncl` to continue to work properly after LAN failure, even when a standby LAN has been configured for the failed primary. These steps also protect against the loss of DNS services, allowing cluster nodes to continue communicating with one another.

1. Edit the `/etc/hosts` file on all nodes in the cluster. Add name resolution for all heartbeat IP addresses, and other IP addresses from all the cluster nodes. Example:

```
15.13.172.231   hasupt01
192.2.1.1       hasupt01
192.2.8.2       hasupt01
15.13.172.232   hasupt02
192.2.1.2       hasupt02
192.2.8.2       hasupt02
15.13.172.233   hasupt03
192.2.1.3       hasupt03
192.2.8.3       sgsupt03
```

   This ensures that messages coming from non-public networks, as well as public networks, are mapped to the correct host name.

**NOTE** For each cluster node, the public network IP address must be the first address listed. This enables other applications to talk to other nodes on public networks.

2. Edit or create the `/etc/nsswitch.conf` file on all nodes and add the following line if it does not already exist:

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

If a line beginning with the string "hosts:" already exists, then make sure that the text immediately to the right of this string is:

```
files [NOTFOUND=continue] dns
```

This step is critical so that the nodes in the cluster can still resolve hostnames to IP addresses while DNS is down or if the primary LAN is down.

3. If not cluster exists on a node, crate and edit an /etc/cmclnodelist file on all nodes and add access to all cluster node primary IP addresses and node names:

```
15.13.172.231       hasupt01
15.13.172.232       hasupt02
15.13.172.233       hasupt03
```

## Creating Mirrors of Root Logical Volumes

It is highly recommended that you use mirrored root volumes on all cluster nodes. The following procedure assumes that you are using separate boot and root volumes; you create a mirror of the boot volume (/dev/vg00/lvol1), primary swap (/dev/vg00/lvol2), and root volume (/dev/vg00/lvol3). In this example and in the following commands, /dev/dsk/c4t5d0 is the primary disk and /dev/dsk/c4t6d0 is the mirror; be sure to use the correct device file names for the root disks on your system.

1. Create a bootable LVM disk to be used for the mirror.

```
# pvcreate -B /dev/rdsk/c4t6d0
```

2. Add this disk to the current root volume group.

```
# vgextend /dev/vg00 /dev/dsk/c4t6d0
```

3. Make the new disk a boot disk.

```
# mkboot -l /dev/rdsk/c4t6d0
```

4. Mirror the boot, primary swap, and root logical volumes to the new bootable disk. Ensure that all devices in vg00, such as /usr, /swap, etc., are mirrored.

**NOTE** The boot, root, and swap logical volumes *must* be done in exactly the following order to ensure that the boot volume occupies the first contiguous set of extents on the new disk, followed by the swap and the root.

The following is an example of mirroring the boot logical volume:

```
# lvextend -m 1 /dev/vg00/lvol1 /dev/dsk/c4t6d0
```

The following is an example of mirroring the primary swap logical volume:

```
# lvextend -m 1 /dev/vg00/lvol2 /dev/dsk/c4t6d0
```

The following is an example of mirroring the root logical volume:

```
# lvextend -m 1 /dev/vg00/lvol3 /dev/dsk/c4t6d0
```

5. Update the boot information contained in the BDRA for the mirror copies of boot, root and primary swap.

```
# /usr/sbin/lvlnboot -b /dev/vg00/lvol1
# /usr/sbin/lvlnboot -s /dev/vg00/lvol2
# /usr/sbin/lvlnboot -r /dev/vg00/lvol3
```

6. Verify that the mirrors were properly created.

```
# lvlnboot -v
```

The output of this command is shown in a display like the following:

```
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
         /dev/dsk/c4t5d0 (10/0.5.0) -- Boot Disk
         /dev/dsk/c4t6d0 (10/0.6.0) -- Boot Disk
Boot:  lvol1    on:      /dev/dsk/c4t5d0
                         /dev/dsk/c4t6d0
Root:  lvol3    on:      /dev/dsk/c4t5d0
                         /dev/dsk/c4t6d0
Swap:  lvol2    on:      /dev/dsk/c4t5d0
                         /dev/dsk/c4t6d0
Dump:  lvol2    on:      /dev/dsk/c4t6d0, 0
```

## Choosing Cluster Lock Disks

The following guidelines apply if you are using a lock disk. The cluster lock disk is configured on a volume group that is physically connected to all cluster nodes. This volume group may also contain data that is used by packages.

When you are using dual cluster lock disks, it is required that the default IO timeout values are used for the cluster lock physical volumes. Changing the IO timeout values for the cluster lock physical volumes can prevent the nodes in the cluster from detecting a failed lock disk within the allotted time period which can prevent cluster re-formations from succeeding. To view the existing IO timeout value, run the following command:

```
# pvdisplay <lock device file name>
```

The IO Timeout value should be displayed as “default.” To set the IO Timeout back to the default value, run the command:

```
# pvchange -t 0 <lock device file name>
```

The use of a dual cluster lock is only allowed with certain specific configurations of hardware. Refer to the discussion in Chapter 3 on “Dual Cluster Lock.”

### Backing Up Cluster Lock Disk Information

After you configure the cluster and create the cluster lock volume group and physical volume, you should create a backup of the volume group configuration data on each lock volume group. Use the `vgcfgbackup` command for each lock volume group you have configured, and save the backup file in case the lock configuration must be restored to a new disk with the `vgcfgrestore` command following a disk failure.

**NOTE** You must use the `vgcfgbackup` and `vgcfgrestore` commands to back up and restore the lock volume group configuration data regardless of how you create the lock volume group.

## Ensuring Consistency of Kernel Configuration

Make sure that the kernel configurations of all cluster nodes are consistent with the expected behavior of the cluster during failover. In particular, if you change any kernel parameters on one cluster node, they may also need to be changed on other cluster nodes that can run the same packages.

## Enabling the Network Time Protocol

It is strongly recommended that you enable network time protocol (NTP) services on each node in the cluster. The use of NTP, which runs as a daemon process on each system, ensures that the system time on all nodes is consistent, resulting in consistent timestamps in log files and consistent behavior of message services. This ensures that applications running in the cluster are correctly synchronized. The NTP services daemon, *xntpd*, should be running on all nodes before you begin cluster configuration. The NTP configuration file is `/etc/ntp.conf`.

For information about configuring NTP services, refer to the chapter "Configuring NTP," in the HP-UX manual, *Installation and Administration of Internet Services*.

## Tuning Network and Kernel Parameters

Serviceguard and its extension products such as SGeSAP, SGeRAC, and SGeFF, have been tested with default values of the supported network and kernel parameters in the `ndd` and `kmtune` utilities.

Adjust these parameters with care.

If you experience problems, return the parameters to their default values. When contacting HP support for any issues regarding Serviceguard and networking, please be sure to share all information about any parameters that were changed from the defaults.

Third-party applications that are running in a Serviceguard environment may require tuning of network and kernel parameters:

- `ndd` is the network tuning utility. For more information, see the man page for `ndd`(1M)
- `kmtune` is the system tuning utility. For more information, see the man page for `kmtune`(1M).

Serviceguard has also been tested with non-default values for these two network parameters:

- ip6_nd_dad_solicit_count - This network parameter enables the Duplicate Address Detection feature for IPv6 address. For more information, see “IPv6 Relocatable Address and Duplicate Address Detection Feature” on page 431 of this manual.
- tcp_keepalive_interval - This network parameter controls the length of time the node will allow an unused network socket to exist before reclaiming its resources so they can be reused.

  Serviceguard supports the tcp_keepalive_interval being changed in the following configurations:

  — Supported with Serviceguard A.11.14 or later.

  — Supported on nodes running HP-UX 11.11 only.

  The following requirements must also be met:

  — The maximum value for tcp_keepalive_interval is 7200000 (2 hours, the HP-UX default value).

  — The minimum value for tcp_keepalive_interval is 60000 (60 seconds).

  — The tcp_keepalive_interval value must be set on a node before Serviceguard is started on that node. This can be done by configuring the new tcp_keepalive_interval in the /etc/rc.config.d/nddconf file, which will automatically set any ndd parameters at system boot time.

  — The tcp_keepalive_interval value must be the same for all nodes in the cluster.

For more information, see “Tunable Kernel Parameters” and “Transport Administrator’s Guide posted at http://docs.hp.com. Click “Browse by Release” then choose your operating system.

## Preparing for Changes in Cluster Size

If you intend to add additional nodes to the cluster online, while it is running, ensure that they are connected to the same heartbeat subnets and to the same lock disks as the other cluster nodes. In selecting a cluster lock configuration, be careful to anticipate any potential need for additional cluster nodes. Remember that a cluster of more than four

nodes *may not use* a lock disk, but a two-node cluster *must* use a cluster lock. Thus, if you will eventually need five nodes, you should build an initial configuration that uses a quorum server.

If you intend to remove a node from the cluster configuration while the cluster is running, ensure that the resulting cluster configuration will still conform to the rules for cluster locks described above.

To facilitate moving nodes in and out of the cluster configuration, you can use SCSI cables with inline terminators, which allow a node to be removed from the bus without breaking SCSI termination. See the section “Online Hardware Maintenance with In-line SCSI Terminator” in the “Troubleshooting” chapter for more information on inline SCSI terminators.

If you are planning to add a node online, and a package will run on the new node, ensure that any existing cluster bound volume groups for the package have been imported to the new node. Also, ensure that the `MAX_CONFIGURED_PACKAGES` parameter is set high enough to accommodate the total number of packages you will be using.

# Setting up the Quorum Server

The quorum server software, which has to be running during cluster configuration, must be installed on a system other than the nodes on which your cluster will be running.

**NOTE**

It is recommended that the node on which the quorum server 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.

## Installing the Quorum Server

Use the HP-UX `swinstall` command to install the quorum server, product number B8467BA, on the system or systems where it will be running. The quorum server is installed outside of the cluster that it serves. The quorum server should be installed outside of the cluster that it serves. More details on installation are found in the *Quorum Server Release Notes* for your version of Quorum Server.

The quorum server executable file, `qs`, is installed in the `/usr/lbin` directory. 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 `/etc/cmcluster/qs_authfile`. Add to the file the names of all cluster nodes that will access cluster services from this quorum server. Use one line per node, as in the following example:

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

To allow access by all nodes, enter plus (**+**), then press the Enter key.

## 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 create a directory `/var/adm/qs`, then redirect `stdout` and `stderr` to a file in this directory, for example, `/var/adm/qs/qs.log`.

You must have root permission to execute the quorum server. On a single system, 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:

```
qs:345:respawn:/usr/lbin/qs >> /var/adm/qs/qs.log 2>&1
```

Start the quorum server as follows:

```
# init q
```

When the command is complete, the prompt appears.

Verify the quorum server is running by checking the `qs.log` file.

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

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

```
Oct 04 12:25:06:0:main:Starting Quorum Server
Oct 04 12:25:09:0:main: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.

## Installing Serviceguard

Installing Serviceguard includes updating the software via Software Distributor. It is assumed that you have already installed HP-UX.

Use the following steps *for each node*:

1. Mount the distribution media in the tape drive or CD ROM reader.
2. Run Software Distributor, using the `swinstall` command.
3. Specify the correct input device.
4. Choose the following bundle from the displayed list:

   ```
   B3935DA                    Serviceguard
   ```

5. After choosing the bundle, select OK. The software is loaded.

For details about running `swinstall` and for creating new user accounts, refer to the HP-UX guide, *Managing Systems and Workgroups*.

# Creating a Storage Infrastructure with LVM

In addition to configuring the cluster, you create the appropriate logical volume infrastructure to provide access to data from different nodes. This is done with Logical Volume Manager (LVM), VERITAS Cluster Volume Manager (CVM), or VERITAS Volume Manager (VxVM). You can also use a mixture of volume types, depending on your needs. LVM and VxVM configuration are done before cluster configuration, and CVM configuration is done after cluster configuration.

This section describes storage configuration with LVM. Separate procedures are given for the following:

- Creating Volume Groups for Mirrored Individual Disks
- Creating Volume Groups for Disk Arrays Using PV Links
- Distributing Volume Groups to Other Nodes

The Event Monitoring Service HA Disk Monitor provides the capability to monitor the health of LVM disks. If you intend to use this monitor for your mirrored disks, you should configure them in physical volume groups. For more information, refer to the manual *Using High Availability Monitors*.

## Creating Volume Groups for Mirrored Individual Data Disks

The procedure described in this section uses **physical volume groups** for mirroring of individual disks to ensure that each logical volume is mirrored to a disk on a different I/O bus. This kind of arrangement is known as **PVG-strict mirroring**. It is assumed that your disk hardware is already configured in such a way that a disk to be used as a mirror copy is connected to each node on a different bus than the bus that is used for the other (primary) copy.

For more information on using LVM, refer to the HP-UX *Managing Systems and Workgroups* manual.

### Using SAM to Create Volume Groups and Logical Volumes

You can use SAM to prepare the volume group and logical volume structure needed for HA packages. In SAM, choose the "Disks and File Systems Area." Then use the following procedure for each volume group and file system you are using with the package:

1. Select the Volume Groups subarea.
2. From the Actions menu, choose Create or Extend.
3. Choose the first physical disk you wish to use in the volume group by selecting it from the list of available disks.
4. Enter the volume group name, e.g., `vgdatabase`.
5. Choose Create or Extend Volume Group.
6. Choose Add New Logical Volumes.
7. When adding logical volumes to the volume group, ensure that you are creating mirrored logical volumes with PVG strict allocation.
8. Specify the file system that is to be mounted on the volume group, for example, `/mnt1`.
9. Repeat the procedure for additional volume groups, logical volumes, and file systems.

Skip ahead to the section "Deactivating the Volume Group".

### Using LVM Commands to Create Volume Groups and Logical Volumes

If your volume groups have not been set up, use the procedure in the next sections. If you have already done LVM configuration, skip ahead to the section "Configuring the Cluster."

**Selecting Disks for the Volume Group** Obtain a list of the disks on both nodes and identify which device files are used for the same disk on both. Use the following command on each node to list available disks as they are known to each system:

```
# lssf /dev/dsk/*
```

In the following examples, we use /dev/rdsk/c1t2d0 and /dev/rdsk/c0t2d0, which happen to be the device names for the same disks on both *ftsys9* and *ftsys10*. In the event that the device file names are different on the different nodes, make a careful note of the correspondences.

**Creating Physical Volumes** On the configuration node (ftsys9), use the `pvcreate` command to define disks as physical volumes. This only needs to be done on the configuration node. Use the following commands to create two physical volumes for the sample configuration:

```
# pvcreate -f /dev/rdsk/c1t2d0
# pvcreate -f /dev/rdsk/c0t2d0
```

**Creating a Volume Group with PVG-Strict Mirroring** Use the following steps to build a volume group on the configuration node (ftsys9). Later, the same volume group will be created on other nodes.

1. First, set up the group directory for vgdatabase:

   ```
   # mkdir /dev/vgdatabase
   ```

2. Next, create a control file named *group* in the directory /dev/vgdatabase, as follows:

   ```
   # mknod /dev/vgdatabase/group c 64 0xhh0000
   ```

   The major number is always 64, and the hexadecimal minor number has the form

   ```
   0xhh0000
   ```

   where *hh* must be unique to the volume group you are creating. Use a unique minor number that is available across all the nodes for the `mknod` command below. (This will avoid further reconfiguration later, when NFS-mounted logical volumes are created in the VG.)

   Use the following command to display a list of existing volume groups:

   ```
   # ls -l /dev/*/group
   ```

3. Create the volume group and add physical volumes to it with the following commands:

   ```
   # vgcreate -g bus0 /dev/vgdatabase /dev/dsk/c1t2d0
   # vgextend -g bus1 /dev/vgdatabase /dev/dsk/c0t2d0
   ```

   The first command creates the volume group and adds a physical volume to it in a physical volume group called *bus0*. The second command adds the second drive to the volume group, locating it in a different physical volume group named *bus1*. The use of physical volume groups allows the use of PVG-strict mirroring of disks and PV links.

4. Repeat this procedure for additional volume groups.

### Creating Logical Volumes

Use the following command to create logical volumes (the example is for /dev/vgdatabase):

```
# lvcreate  -L 120 -m 1 -s g /dev/vgdatabase
```

This command creates a 120 MB mirrored volume named *lvol1*. The name is supplied by default, since no name is specified in the command. The *-s g* option means that mirroring is PVG-strict, that is, the mirror copies of data will be in different physical volume groups.

**NOTE** If you are using disk arrays in RAID 1 or RAID 5 mode, omit the `-m 1` and `-s g` options.

### Creating File Systems

If your installation uses file systems, create them next. Use the following commands to create a file system for mounting on the logical volume just created:

1. Create the file system on the newly created logical volume:

   ```
   # newfs -F vxfs /dev/vgdatabase/rlvol1
   ```

   Note the use of the raw device file for the logical volume.

2. Create a directory to mount the disk:

   ```
   # mkdir /mnt1
   ```

3. Mount the disk to verify your work:

   ```
   # mount /dev/vgdatabase/lvol1 /mnt1
   ```

   Note the `mount` command uses the block device file for the logical volume.

4. Verify the configuration:

   ```
   # vgdisplay -v /dev/vgdatabase
   ```

## Creating Volume Groups for Disk Arrays Using PV Links

If you are configuring volume groups that use mass storage on HP's HA disk arrays, you should use redundant I/O channels from each node, connecting them to separate ports on the array. Then you can define alternate links (also called PV links) to the LUNs or logical disks you have defined on the array. In SAM, choose the type of disk array you wish to configure, and follow the menus to define alternate links.

The following example shows how to configure alternate links using LVM commands. In the example, the following disk configuration is assumed:

```
8/0.15.0 /dev/dsk/c0t15d0  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 0 */
8/0.15.1 /dev/dsk/c0t15d1  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 1 */
8/0.15.2 /dev/dsk/c0t15d2  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 2 */
8/0.15.3 /dev/dsk/c0t15d3  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 3 */
8/0.15.4 /dev/dsk/c0t15d4  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 4 */
8/0.15.5 /dev/dsk/c0t15d5  /* I/O Channel 0 (8/0) SCSI address
 15 LUN 5 */

10/0.3.0 /dev/dsk/c1t3d0   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 0 */
10/0.3.1 /dev/dsk/c1t3d1   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 1 */
10/0.3.2 /dev/dsk/c1t3d2   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 2 */
10/0.3.3 /dev/dsk/c1t3d3   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 3 */
10/0.3.4 /dev/dsk/c1t3d4   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 4 */
10/0.3.5 /dev/dsk/c1t3d5   /* I/O Channel 1 (10/0) SCSI addres
s 3 LUN 5 */
```

Assume that the disk array has been configured, and that both the following device files appear for the same LUN (logical disk) when you run the `ioscan` command:

```
/dev/dsk/c0t15d0
/dev/dsk/c1t3d0
```

Use the following steps to configure a volume group for this logical disk:

1. First, set up the group directory for vgdatabase:

   ```
   # mkdir /dev/vgdatabase
   ```

2. Next, create a control file named *group* in the directory /dev/vgdatabase, as follows:

   ```
   # mknod /dev/vgdatabase/group c 64 0xhh0000
   ```

   The major number is always 64, and the hexadecimal minor number has the form

   ```
   0xhh0000
   ```

   where *hh* must be unique to the volume group you are creating. Use a unique number that is available across all the nodes. (This will avoid further reconfiguration later, when NFS-mounted logical volumes will be created in the VG.)

   Use the following command to display a list of existing group files:

   ```
   # ls -l /dev/*/group
   ```

3. Use the `pvcreate` command on one of the device files associated with the LUN to define the LUN to LVM as a physical volume.

   ```
   # pvcreate /dev/rdsk/c0t15d0
   ```

   It is only necessary to do this with *one* of the device file names for the LUN.

4. Use the following commands to create the volume group itself:

   ```
   # vgcreate /dev/vgdatabase /dev/dsk/c0t15d0
   # vgextend /dev/vgdatabase /dev/dsk/c1t3d0
   ```

You can now use the `vgdisplay -v` command to see the primary and alternate links. LVM will now recognize the I/O channel represented by /dev/dsk/c0t15d0 as the primary link to the disk; if the primary link fails, LVM will automatically switch to the alternate I/O channel represented by /dev/dsk/c1t3d0.

To create logical volumes, use the procedure described in the previous section, “Creating Logical Volumes.”

## Distributing Volume Groups to Other Nodes

After creating volume groups for cluster data, you must make them available to any cluster node that will need to activate the volume group. The cluster lock volume group must be made available to all nodes.

### Deactivating the Volume Group

At the time you create the volume group, it is active on the configuration node (*ftsys9*, for example). Before setting up the volume group for use on other nodes, you must first unmount any file systems that reside on the volume group, then deactivate it. At run time, volume group activation and file system mounting are done through the package control script.

Continuing with the example presented in earlier sections, do the following on *ftsys9*:

```
# umount /mnt1
# vgchange -a n /dev/vgdatabase
```

### Distributing the Volume Group with LVM Commands

Use the following commands to set up the same volume group on another cluster node. In this example, the commands set up a new volume group on *ftsys10* which will hold the same physical volume that was available on *ftsys9*. You must carry out the same procedure separately for each node on which the volume group's package can run.

To set up the volume group on ftsys10, use the following steps:

1. On *ftsys9*, copy the mapping of the volume group to a specified file.

   ```
   # vgexport -p -s -m /tmp/vgdatabase.map  /dev/vgdatabase
   ```

2. Still on *ftsys9*, copy the map file to *ftsys10*:

   ```
   # rcp /tmp/vgdatabase.map ftsys10:/tmp/vgdatabase.map
   ```

3. On *ftsys10*, create the volume group directory:

   ```
   # mkdir /dev/vgdatabase
   ```

4. Still on ftsys10, create a control file named *group* in the directory /dev/vgdatabase, as follows:

   ```
   # mknod /dev/vgdatabase/group c 64 0xhh0000
   ```

   Use the same minor number as on *ftsys9*. Use the following command to display a list of existing volume groups:

```
# ls -l /dev/*/group
```

5. Import the volume group data using the map file from node ftsys9. On node fsys10, enter:

   ```
   # vgimport -s -m /tmp/vgdatabase.map /dev/vgdatabase
   ```

   Note that the disk device names on ftsys10 may be different from their names on ftsys9. You should check to ensure that the physical volume names are correct throughout the cluster.

   When the VG can be activated on this node, perform a vgcfgbackup in the unlikely event that a vgcfgrestore must be performed on this node because of a disaster on the primary node and an LVM problem with the volume group. Do this as shown in the example below:

   ```
   # vgchange -a y /dev/vgdatabase
   # vgcfgbackup /dev/vgdatabase
   # vgchange -a n /dev/vgdatabase
   ```

6. If you are using mirrored individual disks in physical volume groups, check the /etc/lvmpvg file to ensure that each physical volume group contains the correct physical volume names for *ftsys10*.

**NOTE**

When you use PVG-strict mirroring, the physical volume group configuration is recorded in the `/etc/lvmpvg` file on the configuration node. This file defines the physical volume groups which are the basis of mirroring and indicate which physical volumes belong to each PVG. Note that on each cluster node, the `/etc/lvmpvg` file must contain the correct physical volume names for the PVG's disks *as they are known on that node*. Physical volume names for the same disks may not be the same on different nodes. After distributing volume groups to other nodes, you must ensure that each node's /etc/lvmpvg file correctly reflects the contents of all physical volume groups on that node. Refer to the following section, "Making Physical Volume Group Files Consistent."

7. Make sure that you have deactivated the volume group on *ftsys9*. Then enable the volume group on *ftsys10*:

   ```
   # vgchange -a y /dev/vgdatabase
   ```

8. Create a directory to mount the disk:

   ```
   # mkdir /mnt1
   ```

9. Mount and verify the volume group on ftsys10:

```
# mount /dev/vgdatabase/lvol1 /mnt1
```

10. Unmount the volume group on *ftsys10*:

```
# umount /mnt1
```

11. Deactivate the volume group on *ftsys10*:

```
# vgchange -a n /dev/vgdatabase
```

### Making Physical Volume Group Files Consistent

Skip ahead to the next section if you do not use physical volume groups for mirrored individual disks in your disk configuration.

Different volume groups may be activated by different subsets of nodes within a Serviceguard cluster. In addition, the physical volume name for any given disk may be different on one node than it is on another. For these reasons, you must carefully merge the /etc/lvmpvg files on all nodes so that each node has a complete and consistent view of all cluster-aware disks as well as of its own private (non-cluster-aware) disks. To make merging the files easier, be sure to keep a careful record of the physical volume group names on the volume group planning worksheet (described in the “Planning” chapter).

Use the following procedure to merge files between the configuration node (*ftsys9*) and a new node (*ftsys10*) to which you are importing volume groups:

1. Copy `/etc/lvmpvg` from *ftsys9* to `/etc/lvmpvg.new` on *ftsys10*.
2. If there are volume groups in `/etc/lvmpvg.new` that do not exist on *ftsys10*, remove all entries for that volume group from `/etc/lvmpvg.new`.
3. If `/etc/lvmpvg` on *ftsys10* contains entries for volume groups that do not appear in `/etc/lvmpvg.new`, then copy all PVG entries for that volume group to `/etc/lvmpvg.new`.
4. Adjust any physical volume names in `/etc/lvmpvg.new` to reflect their correct names on *ftsys10*.
5. On *ftsys10*, copy `/etc/lvmpvg` to `/etc/lvmpvg.old` to create a backup. Copy `/etc/lvmvpg.new` to `/etc/lvmpvg` on *ftsys10*.

## Creating Additional Volume Groups

The foregoing sections show in general how to create volume groups and logical volumes for use with Serviceguard. Repeat the procedure for as many volume groups as you need to create, substituting other volume group names, logical volume names, and physical volume names. Pay close attention to the disk device names. For example, `/dev/dsk/c0t2d0` on one node may not be `/dev/dsk/c0t2d0` on another node.

# Creating a Storage Infrastructure with VxVM

In addition to configuring the cluster, you create the appropriate logical volume infrastructure to provide access to data from different nodes. This is done with Logical Volume Manager (LVM), VERITAS Volume Manager (VxVM), or VERITAS Cluster Volume Manager (CVM). You can also use a mixture of volume types, depending on your needs. LVM and VxVM configuration are done before cluster configuration, and CVM configuration is done after cluster configuration.

For a discussion of migration from LVM to VxVM storage, refer to Appendix G.

This section shows how to configure new storage using the command set of the VERITAS Volume Manager (VxVM). Once you have created the root disk group (described next), you can use VxVM commands or the Storage Administrator GUI, `vmsa`, to carry out configuration tasks. If you are using `vmsa`, be sure the Storage Administrator server is running before you launch the GUI. Details are given in the *VERITAS Volume Manager for HP-UX Release Notes*. For more information, refer to the *VERITAS VMSA Administrator's Guide*. If you are using commands, refer to the VxVM man pages.

## Initializing the VERITAS Volume Manager

If you are about to create disk groups for the first time, you need to initialize the Volume Manager. This is done by creating a disk group known as *rootdg* that contains at least one disk. Use the following command *once only*, immediately after installing VxVM on each node:

```
# vxinstall
```

This displays a menu-driven program that steps you through the VxVM initialization sequence. From the main menu, choose the "Custom" option, and specify the disk you wish to include in *rootdg*.

**IMPORTANT**

The *rootdg* in the VERITAS Volume Manager is not the same as the HP-UX root disk if an LVM volume group is used for the HP-UX root disk file system. Note also that *rootdg* cannot be used for shared storage. However, *rootdg* can be used for other local filesystems (e.g., `/export/home`), so it need not be wasted.

Note that you should create a root disk group *only once on each node*.

## Converting Disks from LVM to VxVM

You can use the `vxvmconvert(1m)` utility to convert LVM volume groups into VxVM disk groups. Before you can do this, the volume group must be deactivated, which means that any package that uses the volume group must be halted. Follow the conversion procedures outlined in the *VERITAS Volume Manager Migration Guide*. Before you start, be sure to create a backup of each volume group's configuration with the `vgcfgbackup` command, and make a backup of the data in the volume group. Refer also to Appendix H, "Migrating from LVM to VxVM Data Storage" for additional details about conversion.

## Initializing Disks for VxVM

You need to initialize the physical disks that will be employed in VxVM disk groups. To initialize a disk, log on to one node in the cluster, then use the `vxdiskadm` program to initialize multiple disks, or use the `vxdisksetup` command to initialize one disk at a time, as in the following example:

```
# /usr/lib/vxvm/bin/vxdisksetup -i c0t3d2
```

## Initializing Disks Previously Used by LVM

If a physical disk has been previously used with LVM, you should use the `pvremove` command to delete the LVM header data from all the disks in the volume group. In addition, if the LVM disk was previously used in a cluster, you have to re-initialize the disk with the `pvcreate -f` command to remove the cluster ID from the disk.

**NOTE** These commands make the disk and its data unusable by LVM, and allow it to be initialized by VxVM. (The commands should only be used if you have previously used the disk with LVM and do not want to save the data on it.)

You can remove LVM header data from the disk as in the following example (note that all data on the disk will be erased):

```
# pvremove /dev/rdsk/c0t3d2
# pvcreate -f /dev/rdsk/c0t3d2
```

Then, use the `vxdiskadm` program to initialize multiple disks for VxVM, or use the `vxdisksetup` command to initialize one disk at a time, as in the following example:

```
# /usr/lib/vxvm/bin/vxdisksetup -i c0t3d2
```

## Creating Disk Groups

Use `vxdiskadm`, or use the `vxdg` command, to create disk groups, as in the following example:

```
# vxdg init logdata c0t3d2
```

Verify the configuration with the following command:

```
# vxdg list
NAME         STATE                ID

rootdg        enabled             971995699.1025.node1
logdata       enabled             972078742.1084.node1
```

## Creating Volumes

Use the `vxassist` command to create logical volumes. The following is an example:

```
# vxassist -g logdata make log_files 1024m
```

This command creates a 1024 MB volume named *`log_files`* in a disk group named logdata. The volume can be referenced with the block device file `/dev/vx/dsk/logdata/log_files` or the raw (character) device file `/dev/vx/rdsk/logdata/log_files`. Verify the configuration with the following command:

```
# vxprint -g logdata
```

The output of this command is shown in the following example:

```
TY  NAME       ASSOC        KSTATE   LENGTH   PLOFFS   STATE
 TUTILO  PUTILO
```

```
v   logdata    fsgen        ENABLED  1024000          ACTIVE
pl  logdata-01 system       ENABLED  1024000          ACTIVE
```

**NOTE** The specific commands for creating mirrored and multi-path storage using VxVM are described in the *VERITAS Volume Manager Reference Guide*.

## Creating File Systems

If your installation uses file systems, create them next. Use the following commands to create a file system for mounting on the logical volume just created:

1. Create the file system on the newly created volume:

   # **newfs -F vxfs /dev/vx/rdsk/logdata/log_files**

2. Create a directory to mount the volume:

   # **mkdir /logs**

3. Mount the volume:

   # **mount /dev/vx/dsk/logdata/log_files /logs**

4. Check to make sure the file system is present, then unmount the file system:

   # **umount /logs**

## Deporting Disk Groups

After creating the disk groups that are to be used by Serviceguard packages, use the following command with each disk group to allow the disk group to be imported by the package control script on several cluster nodes:

# **vxdg deport <DiskGroupName>**

where **<DiskGroupName>** is the name of the disk group that will be activated by the control script.

When all disk groups have been deported, you must issue the following command on all cluster nodes to allow them to access the disk groups:

```
# vxdctl enable
```

## Re-Importing Disk Groups

After deporting disk groups, they are not available for use on the node until they are imported again either by a package control script or with a `vxdg import` command. If you need to import a disk group manually for maintenance or other purposes, you import it, start up all its logical volumes, and mount filesystems as in the following example:

```
# vxdg import dg_01
# vxvol -g dg_01 startall
# mount /dev/vx/dsk/dg_01/myvol /mountpoint
```

**NOTE** Unlike LVM volume groups, VxVM disk groups are *not* entered in the cluster ASCII configuration file, and they are not entered in the package ASCII configuration file.

## Clearimport at System Reboot Time

At system reboot time, the cmcluster RC script does a `vxdisk clearimport` on all disks formerly imported by the system, provided they have the *noautoimport* flag set, and provided they are not currently imported by another running node. The `clearimport` clears the host ID on the disk group, to allow any node that is connected to the disk group to import it when the package moves from one node to another.

Using the clearimport at reboot time allows Serviceguard to clean up following a node failure, for example, a system crash during a power failure. Disks that were imported at the time of the failure still have the node's ID written on them, and this ID must be cleared before the rebooting node or any other node can import them with a package control script.

Note that the `clearimport` is done for disks previously imported with *noautoimport* set on *any* system that has Serviceguard installed, whether it is configured in a cluster or not.

# Configuring the Cluster

This section describes how to define the basic cluster configuration. To do this in Serviceguard Manager, the graphical user interface, read the next section. If you want to use Serviceguard commands, skip ahead to the section entitled "Using Serviceguard Commands to Configure the Cluster."

## Using Serviceguard Manager to Configure the Cluster

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. 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.

There is online Help available at each step to help you make decisions.

Configure your volume groups before configuring the cluster. If you are using a quorum server as the cluster lock, have it running before configuring the cluster.

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.

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.

## Using Serviceguard Commands to Configure 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. Node names must be 31 bytes or less. Here is an example of the command:

```
# cmquerycl -v -C /etc/cmcluster/clust1.config -n ftsys9 -n
ftsys10
```

The example creates an ASCII template file in the default cluster configuration directory, /etc/cmcluster. The ASCII file is partially filled in with the names and characteristics of cluster components on the two nodes *ftsys9* and *ftsys10*. Do not include the domain name when specifying the node name; for example, specify *ftsys9* and not *ftsys9.cup.hp.com*. 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, networks or disks 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 `-k` and `-w` options:

`-k` eliminates some disk probing, and does not return information about potential cluster lock volume groups and lock physical volumes.

`-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. This is the default. The `-w` none option skips network querying. If you have recently checked the networks. this option will save time.

For complete details, refer to the man page on `cmquerycl(1m)`.

### Cluster Configuration Template File

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 cluster1

# 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 either a lock disk
# or a quorum server.
#
# You can use either the quorum server or the lock disk as
# a cluster lock but not both in the same cluster.
#
# 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.

# Lock Disk Parameters.  Use the FIRST_CLUSTER_LOCK_VG and
# FIRST_CLUSTER_LOCK_PV parameters to define a lock disk.
# The FIRST_CLUSTER_LOCK_VG is the LVM volume group that
# holds the cluster lock. This volume group should not be
# used by any other cluster as a cluster lock device.

# 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.
#
# The value of QS_TIMEOUT_EXTENSION will directly effect the amount
# of time it takes for cluster reformation in the event of failure.
# For example, if QS_TIMEOUT_EXTENSION is set to 10 seconds, the cluster
# reformation will take 10 seconds longer than if the QS_TIMEOUT_EXTENSION
# was set to 0. This delay applies even if there is no delay in
# contacting the Quorum Server.  The recommended value for
# QS_TIMEOUT_EXTENSION is 0, which is used as the default
# and the maximum supported value is 30000000 (5 minutes).
#
# 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
# QS_TIMEOUT_EXTENSION 2000000

QS_HOST                 sysman5
QS_POLLING_INTERVAL     300000000
# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.
# NODE_NAME is the specified nodename in the cluster.
# It must match the hostname and both cannot contain full domain name.
# Each NETWORK_INTERFACE, if configured with IPv4 address,
# must have ONLY one IPv4 address entry with it which could
# be either HEARTBEAT_IP or STATIONARY_IP.
# Each NETWORK_INTERFACE, if configured with IPv6 address(es)
# can have multiple IPv6 address entries(up to a maximum of 2,
# only one IPv6 address entry belonging to site-local scope
# and only one belonging to global scope) which must be all
# STATIONARY_IP. They cannot be HEARTBEAT_IP.

NODE_NAME               fresno
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP        15.13.168.91
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0
```

```
# Warning: There are no standby network interfaces for lan0.

NODE_NAME               lodi
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP        15.13.168.94
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0

# Warning: There are no standby network interfaces for lan0.

# Cluster Timing Parameters (microseconds).

# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
# This default setting yields the fastest cluster reformations.
# However, the use 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
# 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

# The FAILOVER_OPTIMIZATION parameter enables Failover Optimization,
# which reduces the time Serviceguard takes for failover. (Failover
# Optimization cannot, however, change the time an application
# needs to shut down or restart.)
#
# There are four requirements:
#   * The Serviceguard Extension for Faster Failover product
#     (SGeFF) must be installed on all cluster nodes.
#   * Only one or two node clusters are supported.
#   * A quorum server must be configured as the tie-breaker.
#   * The cluster must have more than one heartbeat subnet,
#     and neither can be a serial line (RS232).
#
# Other considerations are listed in the SGeFF Release Notes
# and the Serviceguard manual.
#
# You must halt the cluster to change FAILOVER_OPTIMIZATION
# parameter.
#
```

```
# To enable Failover Optimization, set FAILOVER_OPTIMIZATION
# to TWO_NODE.
# The default is NONE.
#
# FAILOVER_OPTIMIZATION   <NONE/TWO_NODE>

FAILOVER_OPTIMIZATION   NONE

# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT      600000000
NETWORK_POLLING_INTERVAL        2000000

# Network Monitor Configuration Parameters.
# The NETWORK_FAILURE_DETECTION parameter determines how LAN card failures are
detec
ted.
# If set to INONLY_OR_INOUT, a LAN card will be considered down when its inbound
# message count stops increasing or when both inbound and outbound
# message counts stop increasing.
# If set to INOUT, both the inbound and outbound message counts must
# stop increasing before the card is considered down.
NETWORK_FAILURE_DETECTION               INOUT

# 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         150


# 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


# List of cluster aware LVM Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# Neither CVM or VxVM Disk Groups should be used here.
# For example:
# VOLUME_GROUP          /dev/vgdatabase
# VOLUME_GROUP          /dev/vg02


# List of OPS Volume Groups.
# Formerly known as DLM Volume Groups, these volume groups
# will be used by OPS or RAC cluster applications via
# the vgchange -a s command. (Note: the name DLM_VOLUME_GROUP
```

```
# is also still supported for compatibility with earlier versions.)
# For example:
# OPS_VOLUME_GROUP              /dev/vgdatabase
# OPS_VOLUME_GROUP              /dev/vg02
```

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 `/etc/cmcluster/clust1.config` file to your requirements, using the data on the cluster 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 a Lock Disk

A cluster lock is required for two node clusters like the one in this example. The lock must be accessible to all nodes and must be powered separately from the nodes. Refer to the section "Cluster Lock" in Chapter 3 for additional information. Enter the lock disk information following the cluster name. The lock disk must be in an LVM volume group that is accessible to all the nodes in the cluster.

The default `FIRST_CLUSTER_LOCK_VG` and `FIRST_CLUSTER_LOCK_PV` supplied in the ASCII template created with `cmquerycl` are the volume group and physical volume name of a disk chosen based on minimum failover time calculations. You should ensure that this disk meets your power wiring requirements. If necessary, choose a disk powered by a circuit which powers *fewer* than half the nodes in the cluster.

To display the failover times of disks, use the `cmquerycl` command, specifying all the nodes in the cluster. Do not include the node's entire domain name; for example, specify *ftsys9* not *ftsys9.cup.hp.com*:

```
# cmquerycl -v -n ftsys9 -n ftsys10
```

The output of the command lists the disks connected to each node together with the re-formation time associated with each.

**NOTE** You should *not* configure a second lock volume group or physical volume unless your configuration specifically requires it. See the discussion "Dual Cluster Lock" in the section "Cluster Lock" in Chapter 3.

If your configuration requires you to configure a second cluster lock, enter the following parameters in the cluster configuration file:

```
SECOND_CLUSTER_LOCK_VG /dev/volume-group
SECOND_CLUSTER_LOCK_PV /dev/dsk/block-special-file
```

where the */dev/volume-group* is the name of the second volume group and *block-special-file* is the physical volume name of a lock disk in the chosen volume group. These lines should be added for each node.

### Specifying a Quorum Server

To specify a quorum server instead of a lock disk, use the `-q` option of the cmquerycl command, specifying a Quorum Server host server. Example:

```
# cmquerycl -n ftsys9 -n ftsys10 -q qshost
```

The cluster ASCII file that is generated in this case contains parameters for defining the quorum server. This portion of the file is shown below:

```
# 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
# 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 interval 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
# QS_TIMEOUT_EXTENSION 2000000
```

Enter the QS_HOST, QS_POLLING_INTERVAL and, if desired, a QS_TIMEOUT_EXTENSION.

### Identifying Heartbeat Subnets

The cluster ASCII file includes entries for IP addresses on the heartbeat subnet. It is recommended that you use a dedicated heartbeat subnet, but it is possible to configure heartbeat on other subnets as well, including the data subnet.

The heartbeat must be on an IPv4 subnet and must employ IPv4 addresses. IPv6 heartbeat is *not supported*.

**NOTE** If you are using VERITAS CVM disk groups, you can configure only a *single* heartbeat subnet, which should be a dedicated subnet. Each system on this subnet must have standby LANs configured, to ensure that there is a highly available heartbeat path.

### 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 plus approximately 100KB per package in lockable memory. When selecting a value for MAX_CONFIGURED_PACKAGES, be sure to include the CVM-VxVM-PKG as part of the total in MAX_CONFIGURED_PACKAGES if you will be using VERITAS CVM disk storage.

**NOTE** Remember to tune HP-UX kernel parameters on each node to ensure that they are set high enough for the largest number of packages that will ever run concurrently on that node.

### 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*HEARTBEAT_INTERVAL`. A good rule of thumb is to have at least two or three heartbeats within one `NODE_TIMEOUT`.

### Identifying Serial Heartbeat Connections

If you are using a serial (RS232) line as a heartbeat connection, use the `SERIAL_DEVICE_FILE` parameter and enter the device file name that corresponds to the serial port you are using on each node. Be sure that the serial cable is securely attached during and after configuration.

### Optimization

Serviceguard Extension for Faster Failover (SGeFF) is a separately purchased product. If it is installed, the configuration file will display the parameter to enable it.

SGeFF reduces the time it takes Serviceguard to process a failover. It cannot, however, change the time it takes for packages and applications to gracefully shut down and restart.

SGeFF has requirements for cluster configuration, as outlined in the cluster configuration template file.

For more information, see the Serviceguard Extension for Faster Failover Release Notes posted on `http://www.docs.hp.com/hpux/ha`.

### Access Control Policies

New in Serviceguard Version 11.16, Access Control Policies allow non-root user to use common administrative commands.

Non-root users of Serviceguard Manager, the graphical user interface, need to have a configured access policy to view and to administer Serviceguard clusters, packages and packages. In new configurations, it is a good idea to immediately configure at least one monitor access policy.

Check spelling when entering text, especially when typing wildcards, such as ANY_USER and CLUSTER_MEMBER_NODE. If they are misspelled, Serviceguard will assume they are specific users or nodes. You may not configure the access policy you intended to configure.

A root user on the cluster can create or modify access policies while the cluster is running.

### Adding Volume Groups

Add any LVM volume groups you have configured to the ASCII cluster configuration file, with a separate `VOLUME_GROUP` parameter for each cluster-aware volume group that will be used in the cluster. These volume groups will be initialized with the cluster ID when the `cmapplyconf` command is used. In addition, you should add the appropriate volume group, logical volume and file system information to each package control script that activates a volume group. This process is described in Chapter 6.

**NOTE** If you are using CVM disk groups, they should be configured after cluster configuration is done, using the procedures described in “Creating a Storage Infrastructure with CVM” on page 230. VERITAS disk groups are added to the package configuration file, as described in Chapter 6.

## Verifying the Cluster Configuration

In Serviceguard Manager, click the Check button to verify the configuration.

If you have edited an ASCII cluster configuration file using the command line, use the following command to verify the content of the file:

```
# cmcheckconf -k -v -C /etc/cmcluster/clust1.config
```

Both methods check the following:

- Network addresses and connections.
- Cluster lock connectivity (if you are configuring a lock disk).
- Validity of configuration parameters for the cluster and packages.
- 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 `NODE_TIMEOUT` is at least twice the value of `HEARTBEAT_INTERVAL`.
- The value for `AUTO_START_TIMEOUT` variables is >=0.
- Heartbeat network minimum requirement. The cluster must have one heartbeat LAN configured with a standby, two heartbeat LANs, one heartbeat LAN and an RS232 connection, or one heartbeat network with no local LAN switch, but with a primary 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.
- If a serial (RS-232) heartbeat is configured, there are no more than two nodes in the cluster, and no more than one serial (RS232) port connection per node.
- `VOLUME_GROUP` entries are not currently marked as cluster-aware.
- There is only one heartbeat subnet configured if you are using CVM disk storage.

If the cluster is online, the check also verifies that all the conditions for the specific change in configuration have been met.

**NOTE** Using the `-k` option means that `cmcheckconf` only checks disk connectivity to the LVM disks that are identified in the ASCII file. Omitting the `-k` option (the default behavior) means that `cmcheckconf` tests the connectivity of all LVM disks on all nodes. Using `-k` can result in significantly faster operation of the command.

## Distributing the Binary Configuration File

After specifying all cluster parameters, you 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 cluster lock, 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.

### Distributing the Binary File with Serviceguard Manager

When you have finished entering the information, click Apply.

### Distributing the Binary File on the Command Line

Use the following steps to generate the binary configuration file and distribute the configuration to all nodes in the cluster:

- Activate the cluster lock volume group so that the lock disk can be initialized:

  ```
  # vgchange -a y /dev/vglock
  ```

- Generate the binary configuration file and distribute it:

  ```
  # cmapplyconf -k -v -C /etc/cmcluster/clust1.config
  ```

**NOTE** Using the `-k` option means that `cmapplyconf` only checks disk connectivity to the LVM disks that are identified in the ASCII file. Omitting the `-k` option (the default behavior) means that `cmapplyconf` tests the connectivity of all LVM disks on all nodes. Using `-k` can result in significantly faster operation of the command.

- Deactivate the cluster lock volume group.

  ```
  # vgchange -a n /dev/vglock
  ```

The `cmapplyconf` command creates a binary version of the cluster configuration 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 that the `cmapplyconf` command does not distribute the ASCII configuration file.

**CAUTION** The cluster lock volume group must be activated exactly one node before it applying, and it must be deactivated after the configuration is applied.

The apply will not complete unless the lock volume group is active on one node, but not more than one node.

Be sure to deactivate the cluster lock volume group on the configuration node after `cmapplyconf` is executed.

### Storing Volume Group and Cluster Lock Configuration Data

After configuring the cluster, create a backup copy of the LVM volume group configuration by using the `vgcfgbackup` command for each volume group you have created. If a disk in a volume group must be replaced, you can then restore the disk's metadata by using the `vgcfgrestore` command. The procedure is described under “Replacing Disks” in the “Troubleshooting” chapter.

Be sure to use `vgcfgbackup` for all volume groups, including the cluster lock volume group.

**NOTE**

You *must* use the `vgcfgbackup` command to store a copy of the cluster lock disk's configuration data whether you created the volume group using SAM or using HP-UX commands.

If the cluster lock disk ever needs to be replaced while the cluster is running, you *must* use the `vgcfgrestore` command to restore lock information to the replacement disk. Failure to do this might result in a failure of the entire cluster if all redundant copies of the lock disk have failed and if replacement mechanisms or LUNs have not had the lock configuration restored. (If the cluster lock disk is configured in a disk array, RAID protection provides a redundant copy of the cluster lock data. MirrorDisk/UX does not mirror cluster lock information.)

# Creating a Storage Infrastructure with CVM

In addition to configuring the cluster, you create the appropriate logical volume infrastructure to provide access to data from different nodes. This is done with Logical Volume Manager (LVM), VERITAS Volume Manager (VxVM), or VERITAS Cluster Volume Manager (CVM). You can also use a mixture of volume types, depending on your needs. LVM and VxVM configuration are done before cluster configuration, and CVM configuration is done after cluster configuration.

For a discussion of migration from LVM to VxVM or CVM storage, refer to Appendix H.

This section shows how to configure storage using the command set of the VERITAS Cluster Volume Manager (CVM). Before starting, make sure the directory in which VxVM commands are stored (`/usr/lib/vxvm/bin`) is in your path. Once you have created the root disk group with `vxinstall`, you can use VxVM commands or the VERITAS Storage Administrator GUI, `vmsa`, to carry out configuration tasks. If you are using `vmsa`, be sure the storage administrator server is running before you launch the GUI. Detailed instructions for running `vxinstall` are given in the *VERITAS Volume Manager 3.5 Release Notes*. For more information, refer to the *VERITAS Volume Manager 3.5 Administrator's Guide*.

Separate procedures are given below for:

- Creating a Root Disk Group
- Preparing the Cluster for Use with CVM
- Creating Disk Groups for Shared Storage

For more information, including details about configuration of plexes (mirrors), multipathing, and RAID, refer to the HP-UX documentation for the VERITAS Volume Manager.

## Initializing the VERITAS Volume Manager

If you are about to create disk groups for the first time, you need to initialize the Volume Manager. This is done by creating a disk group known as *rootdg* that contains at least one disk. Use the following command after installing VxVM/CVM on each node:

```
# vxinstall
```

This displays a menu-driven program that steps you through the VxVM/CVM initialization sequence. From the main menu, choose the "Custom" option, and specify the disk you wish to include in *rootdg*.

**IMPORTANT**

The *rootdg* in the VERITAS Volume Manager is not the same as the HP-UX root disk if an LVM volume group is used for the HP-UX root file system (/). Note also that *rootdg* cannot be used for shared storage. However, *rootdg* can be used for other local filesystems (e.g., `/export/home`), so it need not be wasted.

Note that you should create a root disk group *only once on each node*.

## Preparing the Cluster for Use with CVM

In order to use the VERITAS Cluster Volume Manager (CVM), you need a cluster that is running with a special CVM package, called a System Multi-node Package (SMP). This means that the cluster must already be configured and running before you create disk groups.

You cannot configure SMP packages through Serviceguard Manager. Once the SMP is configured, however, you can modify its cluster's configuration if you have root login on one of the cluster nodes. You can view the SMP packages if you have monitor access role. You can halt or start the SMP packages in Serviceguard Manager if you have full-admin access role.

**NOTE**

Cluster configuration is described in the previous section.

To prepare the cluster for CVM disk group configuration, you need to set `MAX_CONFIGURED_PACKAGES` to 1 or greater in the cluster ASCII configuration file, and ensure that only one heartbeat subnet is configured. Then use the following command, which creates the special package that communicates cluster information to CVM:

```
# cmapplyconf -P /etc/cmcluster/cvm/VxVM-CVM-pkg.conf
```

**WARNING** **The VxVM-CVM-pkg.conf file should never be edited.**

After this command completes successfully, you can create disk groups for shared use as described in the following sections. The cluster is now running with a special **system multi-node package** named VxVM-CVM-pkg, which is on all nodes. This package is shown in the following output of the cmviewcl command:

```
CLUSTER      STATUS
example      up

NODE         STATUS       STATE
ftsys7       up           running
ftsys8       up           running
ftsys9       up           running
ftsys10      up           running

SYSTEM_MULTI_NODE_PACKAGES:

PACKAGE      STATUS       STATE
VxVM-CVM-pkg up           running
```

## Starting the Cluster and Identifying the Master Node

If it is not already running, start the cluster, which will activate the special CVM package:

# **cmruncl**

When CVM starts up, it selects a master node, and this is the node from which you must issue the disk group configuration commands. To determine the master node, issue the following command from each node in the cluster:

# **vxdctl -c mode**

One node will identify itself as the master. Create disk groups from this node.

## Initializing Disks for CVM

You need to initialize the physical disks that will be employed in CVM disk groups. If a physical disk has been previously used with LVM, you should use the `pvremove` command to delete the LVM header data from all the disks in the volume group (this is not necessary if you have not previously used the disk with LVM).

To initialize a disk for CVM, log on to the master node, then use the `vxdiskadm` program to initialize multiple disks, or use the `vxdisksetup` command to initialize one disk at a time, as in the following example:

```
# /usr/lib/vxvm/bin/vxdisksetup -i c0t3d2
```

## Creating Disk Groups

The following steps should be used to create disk groups.

1. Use the `vxdg` command to create disk groups. Use the -s option to specify shared mode, as in the following example:

```
# vxdg -s init logdata c0t3d2
```

2. Verify the configuration with the following command:

```
# vxdg list
NAME         STATE                ID

rootdg        enabled             971995699.1025.node1
logdata       enabled,shared      972078742.1084.node2
```

3. Activate the disk group, as follows, before creating volumes:

```
# vxdg -g logdata set activation=ew
```

## Creating Volumes

Use the `vxassist` command to create logical volumes, as in the following example:

```
# vxassist -g logdata make log_files 1024m
```

This command creates a 1024 MB volume named *`log_files`* in a disk group named logdata. The volume can be referenced with the block device file `/dev/vx/dsk/logdata/log_files` or the raw (character) device file `/dev/vx/rdsk/logdata/log_files`.

Verify the configuration with the following command:

```
# vxdg list
```

### Mirror Detachment Policies with CVM

The default CVM disk mirror detachment policy is 'global', which means that as soon as one node cannot see a specific mirror copy (plex), all nodes cannot see it as well. The alternate policy is 'local', which means that if one node cannot see a specific mirror copy, then CVM will deactivate access to the volume for that node only. This policy can be re-set on a disk group basis by using the `vxedit` command, as follows:

```
# vxedit set diskdetpolicy=[global|local] <DiskGroupName>
```

**NOTE** The specific commands for creating mirrored and multi-path storage using CVM are described in the HP-UX documentation for the VERITAS Volume Manager.

## Creating File Systems

If your installation uses file systems, create them next. Use the following commands to create a file system for mounting on the logical volume just created:

1. Create the file system on the newly created volume:

   ```
   # newfs -F vxfs /dev/vx/rdsk/logdata/log_files
   ```

2. Create a directory to mount the volume:

   ```
   # mkdir /logs
   ```

3. Mount the volume:

   ```
   # mount /dev/vx/dsk/logdata/log_files /logs
   ```

4. Check to make sure the file system is present, then unmount it:

   ```
   # umount /logs
   ```

5. Use the following command to deactivate the disk group:

   ```
   # vxdg -g logdata set activation=off
   ```

## Adding Disk Groups to the Package Configuration

After creating units of storage with VxVM commands, you need to specify the CVM disk groups in each package configuration ASCII file. Use one `DISK_GROUP` parameter for each disk group the package will use. You also need to identify the CVM disk groups, file systems, logical volumes, and mount options in the package control script. The package configuration process is described in detail in Chapter 6.

**NOTE** Unlike LVM volume groups, CVM disk groups are *not* entered in the cluster ASCII configuration file.

# 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. You can check configuration and status information using Serviceguard Manager:

- You can see if all configured nodes are running.
- You can check that all configured packages are running, and see what nodes they are running on.
- You can get more information from the property sheets for cluster, nodes, and packages.

When you create or modify a package or cluster configuration, you can start the cluster running and archive its configuration in a Serviceguard Manager (.sgm) file. The data in this file can be compared with later versions of the cluster to understand the changes that are made over time. It will be particularly useful in troubleshooting to compare this file to a problem cluster.

There are several administrative commands you can use through Serviceguard Manager, if the Session Server node and the target node both have Serviceguard version A.11.12 or later installed.

## 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.
- 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.

## Preventing Automatic Activation of Volume Groups

It is important to prevent LVM volume groups that are to be used in packages from being activated at system boot time by the `/etc/lvmrc` file. To ensure that this does not happen, edit the `/etc/lvmrc` file on all nodes. Set `AUTO_VG_ACTIVATE` to 0, then include all the volume groups that are not cluster bound in the `custom_vg_activation` function. Volume groups that will be used by packages should *not* be included anywhere in the file, since they will be activated and deactivated by control scripts.

**NOTE** The root volume group does not need to be included in the `custom_vg_activation` function, since it is automatically activated before the /etc/lvmrc file is used at boot time.

## 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. Automatic cluster start is the preferred way to start a cluster. No action is required by the system administrator.

There are three cases:

- The cluster is not running on any node, all cluster nodes must be reachable, and all must be attempting to start up. In this case, the node attempts to form a cluster consisting of all configured nodes.
- The cluster is already running on at least one node. In this case, the node attempts to join that cluster.
- Neither is true: the cluster is not running on any node, and not all the nodes are reachable and trying to start. In this case, the node will attempt to start for the AUTO_START_TIMEOUT period. If neither of these things becomes true in that time, startup will fail.

To enable automatic cluster start, set the flag `AUTOSTART_CMCLD` to 1 in the `/etc/rc.config.d/cmcluster` file on each node in the cluster; the nodes will then join the cluster at boot time.

Here is an example of the */etc/rc.config.d/cmcluster* file:

```
#*********************  CMCLUSTER  *********************
# Highly Available Cluster configuration
#
# @(#) $Revision: 72.2 $
#
# AUTOSTART_CMCLD:    If set to 1, the node will attempt to
#                      join it's CM cluster automatically when
#                      the system boots.
#                      If set to 0, the node will not attempt
#                      to join it's 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/issue and /etc/motd files may be customized to include cluster-related information.

## Managing a Single-Node Cluster

The number of nodes you will need for your Serviceguard cluster depends on the processing requirements of the applications you want to protect. You may want to configure a single-node cluster to take advantage of Serviceguard's network failure protection.

In a single-node cluster, a cluster lock is not required, since there is no other node in the cluster. The output from the `cmquerycl` command omits the cluster lock 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. For standby LANs, all that is required is the `NETWORK_INTERFACE` keyword with the LAN device name.

### 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 Serviceguard daemon *`cmcld`* 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.

However, you should *not* try to restart Serviceguard, since data corruption might occur if the node were to attempt to start up a new instance of the application that is still running on the 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.

## Deleting the Cluster Configuration

With root login, you can delete a cluster configuration from all cluster nodes by using Serviceguard Manager, or on the command line. The `cmdeleteconf` 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 `/etc/cmcluster/cmclconfig`. It does *not* remove any other files from the `/etc/cmcluster` 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:

```
It is recommended that you do not proceed with the
configuration operation unless you are sure these nodes are
permanently unavailable.Do you want to continue?
```

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 for creating package on the command line:

- Creating the Package Configuration
- Writing the Package Control Script
- Verifying the Package Configuration
- Distributing the Configuration

Each of these tasks is described in a separate section below.

You can also create a package and its control script, in Serviceguard Manager. The guided method will step you through the process. Packages created in Serviceguard Manager can be modified in Serviceguard Manager.

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.

Much of the information on the worksheets is discovered automatically by Serviceguard Manager.

# 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.

You can create a package using Serviceguard Manager or using HP-UX commands and editors. The following section describes Serviceguard Manager configuration. If you are using the Serviceguard command line, skip ahead to the section entitled "Using Serviceguard Commands to Create a Package."

## 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 11.16 and later.

1. Connect to a session server node that has Serviceguard version 11.16 installed. Discover a cluster that has version 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.

See “Configuring in Stages” on page 245.

## Using Serviceguard Commands to Configure a Package

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 /etc/cmcluster directory:

   ```
   # mkdir /etc/cmcluster/pkg1
   ```

   You can use any directory names you wish.

2. Next, generate a package configuration template for the package:

   ```
   # cmmakepkg -p /etc/cmcluster/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 (with 31 bytes or less in the name), 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 you 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.

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:

```
# **********************************************************************
# ****** 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 Serviceguard Extension for RAC 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

# Enter the package type for this package. PACKAGE_TYPE indicates
# whether this package is to run as a FAILOVER or SYSTEM_MULTI_NODE
# package.
#
#    FAILOVER     package runs on one node at a time and if a failure
#                 occurs it can switch to an alternate node.
#
#    SYSTEM_MULTI_NODE
#                 package runs on multiple nodes at the same time.
#                 It can not be started and halted on individual nodes.
#                 Both NODE_FAIL_FAST_ENABLED and AUTO_RUN must be set
#                 to YES for this type of package. All SERVICES must
#                 have SERVICE_FAIL_FAST_ENABLED set to YES.
#
# NOTE: Packages which have a PACKAGE_TYPE of SYSTEM_MULTI_NODE are
#       not failover packages and should only be used for applications
#       provided by Hewlett-Packard.
#
```

```
#       Since SYSTEM_MULTI_NODE packages run on multiple nodes at
#       one time, following parameters are ignored:
#
#          FAILOVER_POLICY
#          FAILBACK_POLICY
#       Since an IP address can not be assigned to more than node at a
#       time, relocatable IP addresses can not be assigned in the
#       package control script for multiple node packages. If
#       volume groups are assigned to multiple node packages they must
#       activated in a shared mode and data integrity is left to the
#       application. Shared access requires a shared volume manager.
#
#
# Examples : PACKAGE_TYPE   FAILOVER (default)
#            PACKAGE_TYPE   SYSTEM_MULTI_NODE
#

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


# 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 LOCAL_LAN_FAILOVER_ALLOWED.
# Possible values are YES and NO.
# The default for LOCAL_LAN_FAILOVER_ALLOWED is YES.  In the event of a
# failure, this permits the cluster software to switch LANs locally
# (transfer to a standby LAN card).  Adjust as necessary.
#
# LOCAL_LAN_FAILOVER_ALLOWED replaces obsolete NET_SWITCHING_ENABLED.

LOCAL_LAN_FAILOVER_ALLOWED      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.
# Enter the timeout, specified in seconds, for the run and halt scripts.
# 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
RUN_SCRIPT_TIMEOUT              NO_TIMEOUT
HALT_SCRIPT
HALT_SCRIPT_TIMEOUT             NO_TIMEOUT

# Enter the names of the storage groups configured for this package.
# Repeat this line as necessary for additional storage groups.
#
# Storage groups are only used with CVM disk groups. Neither
# VxVM disk groups or LVM volume groups should be listed here.
# By specifying a CVM disk group with the STORAGE_GROUP keyword
# this package will not run until the VxVM-CVM-pkg package is
# running and thus the CVM shared disk groups are ready for
# activation.
#
# NOTE: Should only be used by applications provided by
#       Hewlett-Packard.
#
# Example : STORAGE_GROUP  dg01
#           STORAGE_GROUP  dg02
#           STORAGE_GROUP  dg03
#           STORAGE_GROUP  dg04
#

# 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>

# 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.
# The subnet names could be IPv4 or IPv6. The network subnet
# names that are to be monitored for this package could be a mix
# of IPv4 or IPv6 subnet names

#SUBNET


# The keywords RESOURCE_NAME, RESOURCE_POLLING_INTERVAL,
# RESOURCE_START, and RESOURCE_UP_VALUE are used to specify Package
# Resource Dependencies.  To define a package Resource Dependency, a
# RESOURCE_NAME line with a fully qualified resource path name, and
# one or more RESOURCE_UP_VALUE lines are required.  The
# RESOURCE_POLLING_INTERVAL and the RESOURCE_START are optional.
```

```
#
# The RESOURCE_POLLING_INTERVAL indicates how often, in seconds, the
# resource is to be monitored.  It will be defaulted to 60 seconds if
# RESOURCE_POLLING_INTERVAL is not specified.
#
# The RESOURCE_START option can be set to either AUTOMATIC or DEFERRED.
# The default setting for RESOURCE_START is AUTOMATIC.  If AUTOMATIC
# is specified, Serviceguard will start up resource monitoring for
# these AUTOMATIC resources automatically when the node starts up.
# If DEFERRED is selected, Serviceguard will not attempt to start
# resource monitoring for these resources during node start up.  User
# should specify all the DEFERRED resources in the package run script
# so that these DEFERRED resources will be started up from the package
# run script during package run time.
#
# RESOURCE_UP_VALUE requires an operator and a value.  This defines
# the resource 'UP' condition.  The operators are =, !=, >, <, >=,
# and <=, depending on the type of value.  Values can be string or
# numeric.  If the type is string, then only = and != are valid
# operators.  If the string contains whitespace, it must be enclosed
# in quotes.  String values are case sensitive.  For example,
#
#                                       Resource is up when its value is
#                                       --------------------------------
#       RESOURCE_UP_VALUE       = UP                    "UP"
#       RESOURCE_UP_VALUE       != DOWN                 Any value except "DOWN"
#       RESOURCE_UP_VALUE       = "On Course"           "On Course"
#
# If the type is numeric, then it can specify a threshold, or a range to
# define a resource up condition.  If it is a threshold, then any operator
# may be used.  If a range is to be specified, then only > or >= may be used
# for the first operator, and only < or <= may be used for the second operator.
# For example,
#                                       Resource is up when its value is
#                                       --------------------------------
#       RESOURCE_UP_VALUE     = 5               5                   (threshold)
#       RESOURCE_UP_VALUE     > 5.1             greater than 5.1    (threshold)
#       RESOURCE_UP_VALUE     > -5 and < 10     between -5 and 10   (range)
#
# Note that "and" is required between the lower limit and upper limit
# when specifying a range.  The upper limit must be greater than the lower
# limit.  If RESOURCE_UP_VALUE is repeated within a RESOURCE_NAME block, then
# they are inclusively OR'd together.  Package Resource Dependencies may be
# defined by repeating the entire RESOURCE_NAME block.
#
# Example : RESOURCE_NAME               /net/interfaces/lan/status/lan0
```

```
#           RESOURCE_POLLING_INTERVAL   120
#           RESOURCE_START              AUTOMATIC
#           RESOURCE_UP_VALUE           = RUNNING
#           RESOURCE_UP_VALUE           = ONLINE
#
#           Means that the value of resource /net/interfaces/lan/status/lan0
#           will be checked every 120 seconds, and is considered to
#           be 'up' when its value is "RUNNING" or "ONLINE".
## Uncomment the following lines to specify Package Resource Dependencies.
#
#RESOURCE_NAME              <Full_path_name>
#RESOURCE_POLLING_INTERVAL  <numeric_seconds>
#RESOURCE_START             <AUTOMATIC/DEFERRED>
#RESOURCE_UP_VALUE          <op> <string_or_numeric> [and <op> <numeric>]

# 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.
#
# 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
```

- 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 in the cluster 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.
- LOCAL_LAN_FAILOVER_ALLOWED. Enter YES to permit switching of the package IP address to a standby LAN, or NO to keep the package from switching locally.
- RUN_SCRIPT and HALT_SCRIPT. Specify the pathname of the package control script (described in the next section). No default is provided.
- STORAGE_GROUP. Specify the names of any CVM storage groups that will be used by this package. Enter each storage group (CVM disk group) on a separate line. Note that CVM storage groups are *not* entered in the cluster ASCII configuration file.

**NOTE** You should not enter LVM volume groups or VxVM disk groups in this file.

- If your package contains 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.
- 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, and it can be either an IPv4 or an IPv6 subnet. Link-local package IPs are not allowed, hence link-local subnets must not be entered in the package ASCII file.
- NODE_FAIL_FAST_ENABLED parameter. Enter YES or NO.
- To configure monitoring within the package for a registered resource, enter values for the following parameters.
- RESOURCE_NAME. Enter the name of a registered resource that is to be monitored by Serviceguard.
- RESOURCE_POLLING_INTERVAL. Enter the time between attempts to assure that the resource is healthy.

- RESOURCE_UP_VALUE. Enter the value or values that determine when the resource is considered to be up. During monitoring, if a different value is found for the resource, the package will fail.
- RESOURCE_START. The RESOURCE_START option is used to determine when Serviceguard should start up resource monitoring for EMS resources. The RESOURCE_START option can be set to either AUTOMATIC or DEFERRED. If AUTOMATIC is specified, Serviceguard will start up resource monitoring for these resources automatically when the Serviceguard cluster daemon starts up on the node. If resources are configured AUTOMATIC, you do not need to define DEFERRED_RESOURCE_NAME in the package control script.

  If DEFERRED is selected, Serviceguard does not attempt to start resource monitoring for these DEFERRED resources during node start up. However, these DEFERRED resources must be specified in the package control script.

  The following is an example of how to configure DEFERRED and AUTOMATIC resources. In the package configuration file, specify resources as follows:

```
RESOURCE_NAME               /net/interfaces/lan/status/lan0
RESOURCE_POLLING_INTERVAL   60
RESOURCE_START              DEFERRED
RESOURCE_UP_VALUE           = UP

RESOURCE_NAME               /net/interfaces/lan/status/lan1
RESOURCE_POLLING_INTERVAL   60
RESOURCE_START              DEFERRED
RESOURCE_UP_VALUE           = UP

RESOURCE_NAME               /net/interfaces/lan/status/lan2
RESOURCE_POLLING_INTERVAL   60
RESOURCE_START              AUTOMATIC
RESOURCE_UP_VALUE           = UP
```

Access Control Policy is a new feature with Serviceguard version 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, user *`ftsys9`* would have role of package admin for this policy. (This role already includes monitor for the entire cluster.)

Data about Access Control Policies in the package configuration files and the cluster configuration file are later combined in the binary cluster configuration file.

## 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."

# 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. You can use Serviceguard Manager (in Guided Mode) to create the control script as it creates your package configuration. You can also use HP-UX commands to create or modify the package control script.

## Using Serviceguard Manager to Write the Package Control Script

Using the guided method in Serviceguard Manager, a standard control script can be automatically created in the configuration screens. When you click Apply, it is automatically distributed to all the nodes. Click the Preview button to view the script as it is being created and modified.

You can choose to edit the control script yourself if you wish. After that, however, Serviceguard Manager will not be able to read your edits or to modify the script further.

Serviceguard Manager does not distribute other files, such as the HA-NFS toolkit, the ECM toolkit, or Metrocluster scripts; you have to manually distribute such files.

## Using Commands to Write the Package Control Script

Each package must have a separate control script, which must be executable.

For security reasons, it must reside in a directory with the string *cmcluster* in the path. 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. 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.

Use the following procedure to create a control script for the sample package *pkg1*.

First, generate a control script template:

```
# cmmakepkg -s /etc/cmcluster/pkg1/pkg1.sh
```

You may customize the script, as described in the section "Customizing the Package Control Script."

## Creating Packages For Database Products

To coordinate the startup and shutdown of database software with cluster node startup and shutdown, you can use the database template files provided with the separately purchasable Enterprise Cluster Master Toolkit product (B5139DA). These files are found in `/opt/cmcluster/toolkit/DB/`. Separate toolkits are available for Oracle, Informix, and Sybase. In addition to the standard package control script, you use the special script that is provided for the database. To set up these scripts, follow the instructions that appear in the README file provided with each toolkit.

## 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. You need to customize as follows:

- Update the PATH statement to reflect any required paths needed to start your services.
- If you are using LVM, enter the names of volume groups to be activated using the `VG[]` array parameters, and select the appropriate options for the storage activation command, including options for mounting and unmounting filesystems, if desired. Do *not* use the `VXVM_DG[]` or `CVM_DG[]` parameters for LVM volume groups.
- If you are using CVM, enter the names of disk groups to be activated using the `CVM_DG[]` array parameters, and select the appropriate storage activation command, `CVM_ACTIVATION_CMD`. Do *not* use the `VG[]` or `VXVM_DG[]` parameters for CVM disk groups.
- If you are using VxVM disk groups without CVM, enter the names of VxVM disk groups that will be imported using the `VXVM_DG[]` array parameters. Enter one disk group per array element. Do *not* use the `CVM_DG[] or VG[]` parameters for VxVM disk groups without CVM. Also, do *not* specify an activation command.
- If you are using mirrored VxVM disks, specify the mirror recovery option `VXVOL`.

- Add the names of logical volumes and file systems that will be mounted on them.
- Select the appropriate options for the storage activation command (not applicable for basic VxVM disk groups), and also include options for mounting filesystems, if desired.
- Specify the filesystem mount retry and unmount count options.
- If your package uses a large number of volume groups or disk groups or mounts a large number of file systems, consider `increasing the number of concurrent vgchange, mount/umount`, and `fsck` operations. The default of 1 is adequate for most packages.
- Define IP subnet and IP address pairs for your package. IPv4 or IPv6 addresses may be used.
- 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 Serviceguard 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`.

### How Control Scripts Manage VxVM Disk Groups

VxVM disk groups outside CVM are outside the control of the Serviceguard cluster. The package control script uses standard VxVM commands to import and deport these disk groups. (For details on importing and deporting disk groups, refer to the discussion of the *import* and *deport* options in the vxdg man page.)

The control script imports disk groups using the vxdg command with the -tfC options. The -t option specifies that the disk is imported with the *noautoimport* flag, which means that the disk will not be automatically re-imported at boot time. Since disk groups included in the package control script are only imported by Serviceguard packages, they should not be auto-imported.

The -f option allows the disk group to be imported even if one or more disks (a mirror, for example) is not currently available. The -C option clears any existing host ID that might be written on the disk from a prior activation by another node in the cluster. If the disk had been in use on another node which has gone down with a TOC, then its host ID may still be written on the disk, and this needs to be cleared so the new node's ID can be written to the disk. Note that the disk groups are not imported clearing the host ID if the host ID is set and matches a node that is not in a failed state. This is to prevent accidental importation of a disk group on multiple nodes which could result in data corruption.

**WARNING** **Although Serviceguard uses the -C option within the package control script framework, this option should not normally be used from the command line. The "Troubleshooting" section shows some situations where you might need to use -C from the command line.**

The following example shows the command with the same options that are used by the control script:

```
# vxdg -tfC import dg_01
```

This command takes over ownership of all the disks in disk group dg_01, even though the disk currently has a different host ID written on it. The command writes the current node's host ID on all disks in disk group dg_01 and sets the **noautoimport** flag for the disks. This flag prevents a disk group from being automatically re-imported by a node following a reboot. If a node in the cluster fails, the host ID is still written on each

disk in the disk group. However, if the node is part of a Serviceguard cluster then on reboot the host ID will be cleared by the owning node from all disks which have the noautoimport flag set, even if the disk group is not under Serviceguard control. This allows all cluster nodes, which have access to the disk group, to be able to import the disks as part of cluster operation.

The control script also uses the `vxvol startall` command to start up the logical volumes in each disk group that is imported.

## Optimizing for Large Numbers of Storage Units

A set of four variables is provided to allow performance improvement when employing a large number of filesystems or storage groups. For more detail, see the comments in the control script template. The four variables are summarized below

- `CONCURRENT_VGCHANGE_OPERATIONS`—defines a number of parallel LVM volume group activations during package startup as well and deactivations during package shutdown.
- `CONCURRENT_DISKGROUP_OPERATIONS`—defines a number of parallel VxVM disk group activations during package startup and deactivations during package shutdown.
- `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.

You can use the -s option with FSCK_OPT and FS_UMOUNT_OPT parameters for environments that use a large number of filesystems. The `-s` option allows mount/umounts and fscks to be done in parallel. (With the standard 11iv1 (11.11) HP-UX, you need to install patches to get this option.)

## Package Control Script Template File

The following is an excerpt from the package control script template file. The file contains separate sections for activation of VxVM and LVM storage groups:

```
# **********************************************************************
# *                                                                    *
# *        HIGH AVAILABILITY PACKAGE CONTROL SCRIPT (template)         *
# *                                                                    *
# *       Note: This file MUST be edited before it can be used.        *
# *                                                                    *
# **********************************************************************

# 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.

. ${SGCONFFILE:=/etc/cmcluster.conf}

# UNCOMMENT the variables as you set them.

# Set PATH to reference the appropriate directories.
PATH=$SGSBIN:/usr/bin:/usr/sbin:/etc:/bin

# VOLUME GROUP ACTIVATION:
# Specify the method of activation for volume groups.
# Leave the default ("VGCHANGE="vgchange -a e") if you want volume
# groups activated in exclusive mode. This assumes the volume groups have
# been initialized with 'vgchange -c y' at the time of creation.
#
# Uncomment the first line (VGCHANGE="vgchange -a e -q n"), and comment
# out the default, if your disks are mirrored on separate physical paths,
#
# Uncomment the second line (VGCHANGE="vgchange -a e -q n -s"), and comment
# and you want the mirror resynchronization to ocurr in parallel with
# the package startup.
#
# Uncomment the third line (VGCHANGE="vgchange -a y") if you wish to
# use non-exclusive activation mode. Single node cluster configurations
# must use non-exclusive activation.
#
# VGCHANGE="vgchange -a e -q n"
# VGCHANGE="vgchange -a e -q n -s"
# VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"               # Default

# CVM DISK GROUP ACTIVATION:
# Specify the method of activation for CVM disk groups.
```

```
# Leave the default
# (CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=exclusivewrite")
# if you want disk groups activated in the exclusive write mode.
#
# Uncomment the first line
# (CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=readonly"),
# and comment out the default, if you want disk groups activated in
# the readonly mode.
#
# Uncomment the second line
# (CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedread"),
# and comment out the default, if you want disk groups activated in the
# shared read mode.
#
# Uncomment the third line
# (CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedwrite"),
# and comment out the default, if you want disk groups activated in the
# shared write mode.
#
# CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=readonly"
# CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedread"
# CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=sharedwrite"
CVM_ACTIVATION_CMD="vxdg -g \$DiskGroup set activation=exclusivewrite"

# 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.
#
#VG[0]=""

# CVM DISK GROUPS
# Specify which cvm disk groups are used by this package. Uncomment
# CVM_DG[0]="" and fill in the name of your first disk group. You must
# begin with CVM_DG[0], and increment the list in sequence.
#
# For example, if this package uses your disk groups dg01 and dg02, enter:
#         CVM_DG[0]=dg01
#         CVM_DG[1]=dg02
```

```
#
# The cvm disk group activation method is defined above. The filesystems
# associated with these volume groups are specified below in the CVM_*
# variables.
#
#CVM_DG[0]=""

# VxVM DISK GROUPS
# Specify which VxVM disk groups are used by this package. Uncomment
# VXVM_DG[0]="" and fill in the name of your first disk group. You must
# begin with VXVM_DG[0], and increment the list in sequence.
#
# For example, if this package uses your disk groups dg01 and dg02, enter:
#         VXVM_DG[0]=dg01
#         VXVM_DG[1]=dg02
#
# The cvm disk group activation method is defined above.
#
#VXVM_DG[0]=""

#
# NOTE: A package could have LVM volume groups, CVM disk groups and VxVM
#       disk groups.
#
# NOTE: When VxVM is initialized it will store the hostname of the
#       local node in its volboot file in a variable called 'hostid'.
#       The MC Serviceguard package control scripts use both the values of
#       the hostname(1m) command and the VxVM hostid. As a result
#       the VxVM hostid should always match the value of the
#       hostname(1m) command.
#
#       If you modify the local host name after VxVM has been
#       initialized and such that hostname(1m) does not equal uname -n,
#       you need to use the vxdctl(1m) command to set the VxVM hostid
#       field to the value of hostname(1m). Failure to do so will
#       result in the package failing to start.

# FILESYSTEMS
# Filesystems are defined as entries specifying the logical volume, the
# mount point, the mount, umount and fsck options and type of the file system.
# 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 above.
#
```

```
# Specify the filesystems which are used by this package. Uncomment
# LV[0]=””; FS[0]=””; FS_MOUNT_OPT[0]=””; FS_UMOUNT_OPT[0]=””;
FS_FSCK_OPT[0]=””
# FS_TYPE[0]=”” and fill in the name of your first logical volume,
# filesystem, mount, umount and fsck options and filesystem type
# for the file system. You must begin with LV[0], FS[0],
# FS_MOUNT_OPT[0], FS_UMOUNT_OPT[0], FS_FSCK_OPT[0], FS_TYPE[0]
# and increment the list in sequence.
#
# Note: The FS_TYPE parameter lets you specify the type of filesystem to be
# mounted. Specifying a particular FS_TYPE will improve package failover time.

# The FSCK_OPT and FS_UMOUNT_OPT parameters can be used to include the
# -s option with the fsck and umount commands to improve performance for
# environments that use a large number of filesystems. (An example of a
# large environment is given below following the decription of the
# CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS parameter.)
#
# Example: If a package uses two JFS filesystems, pkg01a and pkg01b,
# which are mounted on LVM logical volumes lvol1 and lvol2 for read and
# write operation, you would enter the following:
#      LV[0]=/dev/vg01/lvol1; FS[0]=/pkg01a; FS_MOUNT_OPT[0]=”-o rw”;
#      FS_UMOUNT_OPT[0]=””; FS_FSCK_OPT[0]=””; FS_TYPE[0]=”vxfs”
#
#      LV[1]=/dev/vg01/lvol2; FS[1]=/pkg01b; FS_MOUNT_OPT[1]=”-o rw”
#      FS_UMOUNT_OPT[1]=””; FS_FSCK_OPT[1]=””; FS_TYPE[1]=”vxfs”
#
#LV[0]=””; FS[0]=””; FS_MOUNT_OPT[0]=””; FS_UMOUNT_OPT[0]=””;
FS_FSCK_OPT[0]=””
#FS_TYPE[0]=””
#
# VOLUME RECOVERY
#
# When mirrored VxVM volumes are started during the package control
# bring up, if recovery is required the default behavior is for
# the package control script to wait until recovery has been
# completed.
#
# To allow mirror resynchronization to ocurr in parallel with
# the package startup, uncomment the line
# VXVOL=”vxvol -g \$DiskGroup -o bg startall” and comment out the default.
#
# VXVOL=”vxvol -g \$DiskGroup -o bg startall”
VXVOL=”vxvol -g \$DiskGroup startall”      # Default

# 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 user 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 -ku" to freeup busy mount point.
FS_MOUNT_RETRY_COUNT=0
#

# Configuring the concurrent operations below can be used to improve the
# 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.  For example, if you set CONCURRENT_VGCHANGE_OPERATIONS=5
# and the node where the package is started has only one processor, then
# running concurrent volume group activations will not be beneficial.
# 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 VGCHANGE OPERATIONS
# Specify the number of concurrent volume group activations or
```

```
# deactivations to allow during package startup or shutdown.
# Setting this value to an appropriate number may improve the performance
# while activating or deactivating a large number of volume groups 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_VGCHANGE_OPERATIONS=1

# CONCURRENT DISK GROUP OPERATIONS
# Specify the number of concurrent VxVM DG imports or deports to allow
# during package startup or shutdown.
# Setting this value to an appropriate number may improve the performance
# while importing or deporting a large number of disk groups 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_DISKGROUP_OPERATIONS=1

# CONCURRENT FSCK OPERATIONS
# Specify the number of concurrent fsck 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

# Example:  If a package uses 50 JFS filesystems, pkg01aa through pkg01bx,
# which are mounted on the 50 logical volumes lvol1..lvol50 for read and write
# operation, you may enter the following:
#
#       CONCURRENT_DISKGROUP_OPERATIONS=50
#       CONCURRENT_FSCK_OPERATIONS=50
#       CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS=50
#
#       LV[0]=/dev/vg01/lvol1; FS[0]=/pkg01aa; FS_MOUNT_OPT[0]="-o rw";
#       FS_UMOUNT_OPT[0]="-s"; FS_FSCK_OPT[0]="-s"; FS_TYPE[0]="vxfs"
```

```
#
#      LV[1]=/dev/vg01/lvol2; FS[1]=/pkg01ab; FS_MOUNT_OPT[1]="-o rw"
#      FS_UMOUNT_OPT[1]="-s"; FS_FSCK_OPT[1]="-s"; FS_TYPE[0]="vxfs"
#         :           :          :
#         :           :          :
#         :           :          :
#      LV[49]=/dev/vg01/lvol50; FS[49]=/pkg01bx; FS_MOUNT_OPT[49]="-o rw"
#      FS_UMOUNT_OPT[49]="-s"; FS_FSCK_OPT[49]="-s"; FS_TYPE[0]="vxfs"
#
# IP ADDRESSES
# Specify the IP and Subnet address pairs which are used by this package.
# You could specify IPv4 or IPv6 IP and subnet address pairs.
# 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: Run "netstat -i" to see the available subnets in the Network field.
#
# For example, if this package uses an IPv6 IP of 2001::1/64
# The address prefix identifies the subnet as 2001::/64 which is an
available
# subnet.
# enter:
#          IP[0]=2001::1
#          SUBNET[0]=2001::/64
#          (netmask=ffff:ffff:ffff:ffff::)
# Alternatively the IPv6 IP/Subnet pair can be specified without the prefix
# for the IPv6 subnet.
#          IP[0]=2001::1
#          SUBNET[0]=2001::
#          (netmask=ffff:ffff:ffff:ffff::)
#
# Hint: Run "netstat -i" to see the available IPv6 subnets by looking
# at the address prefixes
# 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]=""
```

```
# 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]=pkg1a
#          SERVICE_CMD[0]="/usr/bin/X11/xclock -display 192.10.25.54:0"
#          SERVICE_RESTART[0]=""  # Will not restart the service.
#
#          SERVICE_NAME[1]=pkg1b
#          SERVICE_CMD[1]="/usr/bin/X11/xload -display 192.10.25.54:0"
#          SERVICE_RESTART[1]="-r 2"   # Will restart the service twice.
#
#          SERVICE_NAME[2]=pkg1c
#          SERVICE_CMD[2]="/usr/sbin/ping"
#          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 /usr/bin/sh.
#
#SERVICE_NAME[0]=""
#SERVICE_CMD[0]=""
#SERVICE_RESTART[0]=""

# DEFERRED_RESOURCE NAME
# Specify the full path name of the 'DEFERRED' resources configured for
# this package.  Uncomment DEFERRED_RESOURCE_NAME[0]="" and fill in the
# full path name of the resource.
#
#DEFERRED_RESOURCE_NAME[0]=""

# DTC manager information for each DTC.
# Example: DTC[0]=dtc_20
#DTC_NAME[0]=
#
#HA_NFS_SCRIPT_EXTENSION
# If the package uses HA NFS, this variable can be used to alter the
# name of the HA NFS script.  If not set, the name of this script is
# assumed to be "ha_nfs.sh".  If set, the "sh" portion of the default
# script name is replaced by the value of this variable.  So if
```

```
# HA_NFS_SCRIPT_EXTENSION is set to "package1.sh", for example, the name
# of the HA NFS script becomes "ha_nfs.package1.sh".  In any case,
# the HA NFS script must be placed in the same directory as the package
# control script.  This allows multiple packages to be run out of the

# same directory, as needed by SGeSAP.
#HA_NFS_SCRIPT_EXTENSION=""
```

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, HP-UX 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.

If you have defined an EMS resource in the package configuration file that is labeled as `DEFERRED`, you need to define a `DEFERRED_RESOURCE_NAME` in the package control script. Specify only the deferred resources, using the `DEFERRED_RESOURCE_NAME` parameter:

```
DEFERRED_RESOURCE_NAME[0]="/net/interfaces/lan/status/lan0"
DEFERRED_RESOURCE_NAME[1]="/net/interfaces/lan/status/lan1"
```

After customizing the script, distribute it to each node in the cluster using `rcp`, `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.

## Support for Additional Products

The package control script template provides exits for use with additional products, including MetroCluster with Continuous Access/CA, MetroCluster with EMC SRDF, and the HA NFS toolkit. Refer to the additional product's documentation for details about how to create a package using the hooks that are provided in the control script.

# Verifying the Package Configuration

Serviceguard automatically checks the configuration you enter and reports any errors.

If Serviceguard Manager created the file, click the Check button or the Apply button.

If you have edited an ASCII package configuration file, use the following command to verify the content of the file:

```
# cmcheckconf -v -P /etc/cmcluster/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.

Serviceguard Manager's Check button and the `cmcheckconf` command check 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.

# Distributing the Configuration

You can use Serviceguard Manger or HP-UX commands to distribute the binary cluster configuration file among the nodes of the cluster.

## Distributing the Configuration File And Control Script with Serviceguard Manager

When you have finished creating a package in Serviceguard Manager, click the Apply button. The binary configuration file will be created and automatically distributed to the cluster nodes. The control script will also be distributed. If you chose the Guided Method, Serviceguard Manger will create the control script will be created; if you edit the control script yourself, it will take your last saved version.

## Copying Package Control Scripts with HP-UX commands

Use HP-UX 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., `rcp` or `ftp`). For example, from *`ftsys9`*, you can issue the `rcp` command to copy the package control script to *`ftsys10`*:

```
# rcp /etc/cmcluster/pkg1/control.sh
ftsys10:/etc/cmcluster/pkg1/control.sh
```

## Distributing the Binary Cluster Configuration File with HP-UX Commands

Use the following steps from the node on which the ASCII cluster and package configuration files exist:

- Verify that the configuration file is correct. Use the following command:

  ```
  # cmcheckconf -C /etc/cmcluster/cmcl.config -P \
    /etc/cmcluster/pkg1/pkg1.config
  ```

- Activate the cluster lock volume group so that the lock disk can be initialized:

```
# vgchange -a y /dev/vg01
```

- Generate the binary configuration file and distribute it across the nodes.

```
# cmapplyconf -v -C /etc/cmcluster/cmcl.config -P \
  /etc/cmcluster/pkg1/pkg1.config
```

- If you are using a lock disk, deactivate the cluster lock volume group.

```
# vgchange -a n /dev/vg01
```

The `cmapplyconf` command creates a binary version of the cluster configuration 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.

## 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.

# 7 Cluster and Package Maintenance

This chapter describes how to see cluster configuration and status information, 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 Status
- Managing the Cluster and Nodes
- Managing Packages and Services
- Reconfiguring a Cluster
- Reconfiguring a Package
- Responding to Cluster Events
- Removing Serviceguard from a System

Most administration tasks can be carried out using Serviceguard Manager or HP-UX commands.

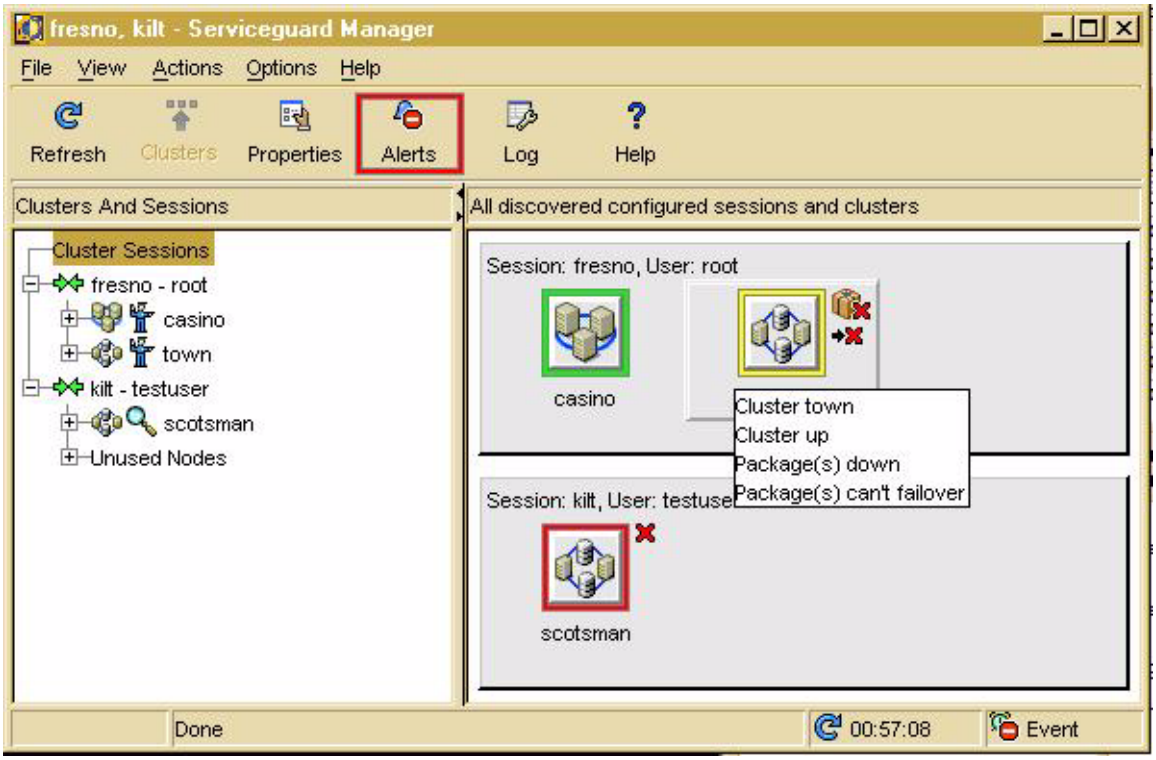
# Reviewing Cluster and Package Status

You can check status using Serviceguard Manager or on a cluster node's command line.

## Reviewing Cluster and Package Status with Serviceguard Manager

Serviceguard Manager shows status several ways.

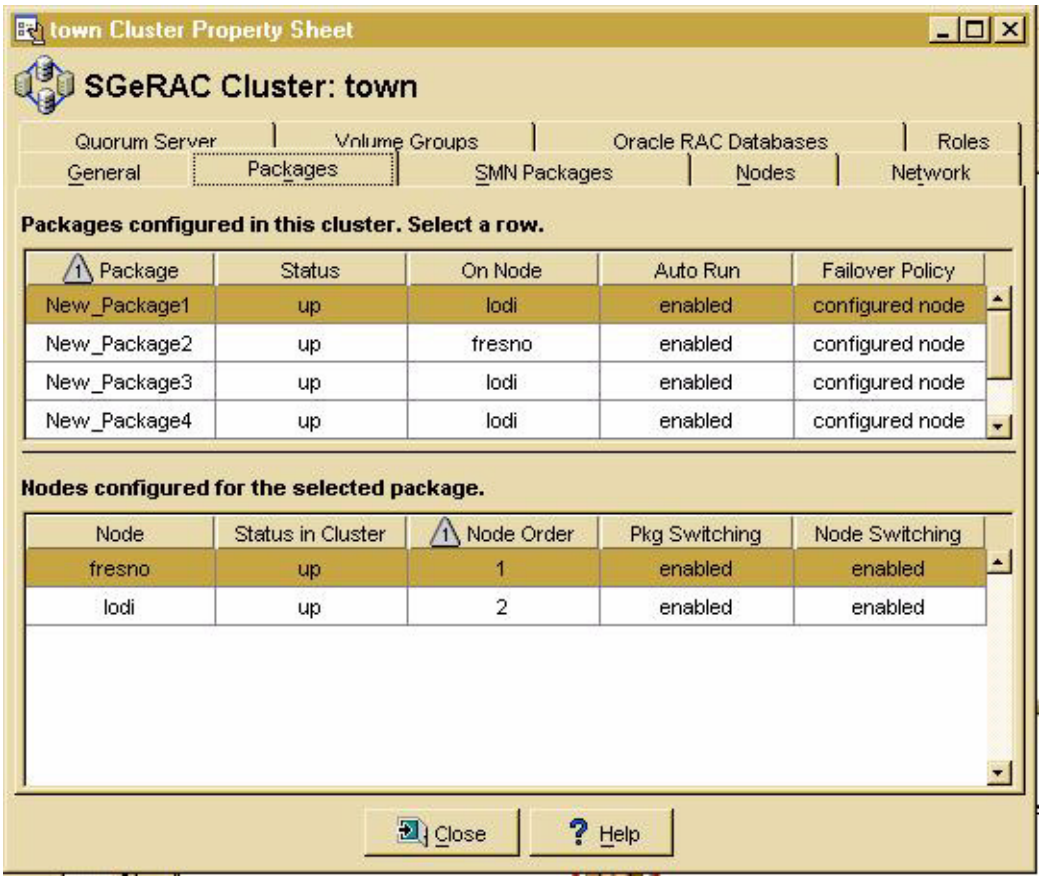
**Figure 7-1 Reviewing Status: Serviceguard Manager Map**




- On the map, cluster object icons have borders to show problems. To the right of the icons, a badge gives information about the type of problem. When you hover the mouse, a popup gives you information.

- There are more details in the cluster, node, and package property sheets.

**Figure 7-2 Reviewing Status: Serviceguard Manager Property Sheet**




## 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 11.16 or later, 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.

You can also specify that the output should be formatted as it was in a specific earlier release by using the -r option indicating the release format you wish. Example:

```
# cmviewcl -r A.11.09
```

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 HP-UX.

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.

**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.

- Switching Enabled for a Node. 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 standby interface has this status.

**Serial Line Status** The serial line has only status, as follows:

- `Up`. Heartbeats are received over the serial line.
- `Down`. Heartbeat has not been received over the serial line within 2 times the `NODE_TIMEOUT` value.
- `Recovering`. A corrupt message was received on the serial line, and the line is in the process of resynchronizing.
- `Unknown`. We cannot determine whether the serial line is up or down. This can happen when the remote node is down.

**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       PATH         NAME
    PRIMARY      up           56/36.1      lan0
    STANDBY      up           60/6         lan1

    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       (cur
rent)
      Alternate    up           enabled      ftsys10

 NODE          STATUS       STATE
  ftsys10      up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           28.1         lan0
    STANDBY      up           32.1         lan1

    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
    pkg2         up           running      enabled      ftsys1
0

      Policy_Parameters:
      POLICY_NAME      CONFIGURED_VALUE
      Failover         configured_node
      Failback         manual

      Script_Parameters:
      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      (cur
rent)
      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
```

**CVM Package Status** If the cluster is using the VERITAS Cluster Volume Manager for disk storage, the system multi-node package CVM-VxVM-pkg must be running on all active nodes for applications to be able to access CVM disk groups. This package is shown in the following output of the `cmviewcl` command:

```
CLUSTER      STATUS
example      up

NODE         STATUS       STATE
ftsys7       down         halted
ftsys8       down         halted
ftsys9       up           running
ftsys10      up           running

SYSTEM_MULTI_NODE_PACKAGES:

PACKAGE      STATUS       STATE
VxVM-CVM-pkg up           running
```

When you use the -v option, the display shows the system multi-node package associated with each active node in the cluster, as in the following:

```
SYSTEM_MULTI_NODE_PACKAGES:

    PACKAGE      STATUS       STATE
    VxVM-CVM-pkg up           running

     NODE         STATUS       STATE
     ftsys7       down         halted

     NODE         STATUS       STATE
     ftsys8       down         halted

     NODE         STATUS       STATE
     ftsys9       up           running
      Script_Parameters:
      ITEM       STATUS   MAX_RESTARTS  RESTARTS   NAME
      Service    up                0          0
VxVM-CVM-pkg.srv

     NODE         STATUS       STATE
     ftsys10      up           running
      Script_Parameters:
      ITEM       STATUS   MAX_RESTARTS  RESTARTS   NAME
      Service    up                0          0
VxVM-CVM-pkg.srv
```

**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       PATH         NAME
    PRIMARY      up           56/36.1      lan0
    STANDBY      up           60/6         lan1

    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       servi
ce1
      Subnet       up                 0          0       15.13
.168.0
      Resource     up                                    /exam
ple/float

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ftsys9       (cur
rent)
      Alternate    up           enabled      ftsys10


  NODE          STATUS       STATE
  ftsys10       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           28.1         lan0
    STANDBY      up           32.1         lan1

UNOWNED_PACKAGES

    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
    pkg2         down         unowned      disabled     unowne
d

      Policy_Parameters:
      POLICY_NAME      CONFIGURED_VALUE
      Failover         configured_node
      Failback         manual


      Script_Parameters:
      ITEM         STATUS    NODE_NAME       NAME
      Resource     down      ftsys9          /example/float
      Subnet       up        ftsys9          15.13.168.0
      Resource     up        ftsys10         /example/float
      Subnet       up        ftsys10         15.13.168.0

      Node_Switching_Parameters:
```

```
      NODE_TYPE    STATUS       SWITCHING    NAME
      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. Resource “/example/float,” which is configured as a dependency of pkg2, is down on one node. 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       PATH         NAME
    PRIMARY      up           56/36.1      lan0
    STANDBY      up           60/6         lan1


    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       servi
ce1
      Subnet       up                 0          0       15.13
.168.0
      Resource     up                                    /exam
```

```
ple/float

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ftsys9       (cur
rent)
      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           service2.1               0
       0
      Subnet       up           15.13.168.0              0
       0
      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ftsys10
      Alternate    up           enabled      ftsys9     (curre
nt)

  NODE         STATUS       STATE
  ftsys10      up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY        up           28.1         lan0
    STANDBY        up           32.1         lan1
```

Now pkg2 is running on node ftsys9. Note that it is still disabled from switching.

**Status After Auto Run is Enabled** The following command changes package switching flag 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 RS232 Status** If you are using a serial (RS232) line as a heartbeat connection, you will see a list of configured RS232 device files in the output of the `cmviewcl -v` command. The following shows normal running status:

```
CLUSTER      STATUS
example      up
  NODE         STATUS       STATE
  ftsys9       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           56/36.1      lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0              up          ftsys10     /dev/tty0
p0
  NODE         STATUS       STATE
  ftsys10       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           28.1         lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0              up          ftsys9
     /dev/tty0p0
```

The following display shows status after node *ftsys10* has halted:

```
CLUSTER      STATUS
example      up
  NODE         STATUS       STATE
  ftsys9       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           56/36.1      lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0            unknown         ftsys10   /dev/tty0
p0
  NODE         STATUS       STATE
  ftsys10      down         running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           28.1         lan0
```

```
    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0            unknown        ftsys9    /dev/tty0p0
```

The following shows status when the serial line is not working:

```
CLUSTER      STATUS
example      up
  NODE          STATUS       STATE
  ftsys9        up           running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            56/36.1       lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0             down          ftsys10   /dev/tty0p0
  NODE          STATUS       STATE
  ftsys10        up           running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            28.1          lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME      STATUS       CONNECTED_TO:
    /dev/tty0p0             down          ftsys9
     /dev/tty0p0
```

**Viewing Data on Unowned Packages** The following example shows packages that are currently unowned, that is, not running on any configured node. Information on monitored resources is provided for each node on which the package can run; this allows you to identify the cause of a failure and decide where to start the package up again.

```
UNOWNED_PACKAGES

PACKAGE      STATUS       STATE        AUTO_RUN     NODE
PKG3         down         halted       disabled     unowned

Policy_Parameters:
POLICY_NAME     CONFIGURED_VALUE
Failover        min_package_node
Failback        automatic

Script_Parameters:
```

```
ITEM       STATUS   NODE_NAME    NAME
Resource   up       manx         /resource/random
Subnet     up       manx         192.8.15.0
Resource   up       burmese      /resource/random
Subnet     up       burmese      192.8.15.0
Resource   up       tabby        /resource/random
Subnet     up       tabby        192.8.15.0
Resource   up       persian      /resource/random
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
```

**Viewing Data on System Multi-Node Packages** The following example shows a cluster that includes system multi-node packages as well as standard Serviceguard packages. The system multi-node packages are running on all nodes in the cluster, whereas the standard packages run on only one node at a time.

```
CLUSTER      STATUS
cats         up

NODE         STATUS       STATE
manx         up           running

    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
    pkg1         up           running      enabled      manx

NODE         STATUS       STATE
tabby        up           running


    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
    pkg2         up           running      enabled      tabby

SYSTEM_MULTI_NODE_PACKAGES:

PACKAGE      STATUS       STATE
VxVM-CVM-pkg up           running
```

# 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

In Serviceguard 11.16 and later, these Package Admin and Cluster Admin commands can be done by non-root users, according to access policies in the cluster's configuration files. See Editing Security Files, in Chapter 5, for more information about configuring access.

You can use Serviceguard Manager or the Serviceguard command line to start or stop the cluster, or to add or halt nodes. 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

You can use Serviceguard Manager or Serviceguard commands to start the cluster.

### Using Serviceguard Manager to Start the Cluster

Select the cluster icon, then right-click to display the action menu. Select "Run cluster *<clustername>*." 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 Serviceguard Commands to Start the Cluster

Use the `cmruncl` 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 command starts all nodes configured in the cluster and verifies the network information:

```
# cmruncl -v
```

By default, cmruncl will do network validation, making sure the actual network setup matches the configured network setup. This is the recommended method. If you have recently checked the network and find the check takes a very long time, you can use the `-w none` option to bypass the validation.

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.)

```
# cmruncl -v -n ftsys9 -n ftsys10
```

**WARNING** **Serviceguard cannot guarantee data integrity if you try to start a cluster with the `cmruncl -n` command while a subset of the cluster's nodes are already running a cluster. If the network connection is down between nodes, using `cmruncl -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 Serviceguard commands to bring a configured node up within a running cluster.

### Using Serviceguard Manager to Add a Configured Node to the Cluster

Select the node icon, then right-click to display the action menu. Select "Run node *<hostname>*." The progress window shows messages as the action takes place. This will also start any packages that are eligible to run on the node. Click OK on the progress window when the operation is complete.

### Using 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
```

By default, cmruncl will do network validation, making sure the actual network setup matches the configured network setup. This is the recommended method. If you have recently checked the network and find the check takes a very long time, you can use the `-w none` option to bypass the validation.

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-UX 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 node *<nodename>*" 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 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 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 cluster *<clustername>*." 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 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 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.

## 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 `/etc/rc.config.d/cmcluster` file.

# Managing Packages and Services

Managing packages and services involves the following tasks:

- Starting a Package
- Halting a Package
- Moving a Package (halt, then start)
- Changing Package Switching Behavior

In Serviceguard 11.16 and later, these commands can be done by non-root users, according to access policies in the cluster's configuration files. See Editing Security Files, in Chapter 5, for more information about configuring access.

You can use Serviceguard Manager or the Serviceguard command line to perform these tasks.

## 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 on the Serviceguard command line.

### 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 "Run package *<pkgname>*" or "Run package *<pkgname>* on node..." with 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 messages for starting the package.

The cluster must be running in order to start a package.

### Using 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 a 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 on the Serviceguard command line. 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 package *<pkgname>*." 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 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 Serviceguard commands to move a package from one node to another.

### Using Serviceguard Manager to Move a Package

The package must be running to start the operation. You can select the package on the map or tree and drag it with your mouse to another cluster node.

Or, select the icon of the package you wish to halt, and right-click to display the action list. Select “Move package to node.” Or, select the package and go to the toolbar menu and choose Actions -> Administering.

The Operation Log 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.

After moving, you may want to change the switching flags. You can do this from the Actions menu, or the right-click popup menu.

### Using 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. `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
```

## Changing Package Switching Behavior

There are two types of switching flags that apply to packages:

- package switching, or the ability to move from one node to another
- node switching, or the ability to move to a specific node

The initial setting for package switching is determined by the `AUTO_RUN` parameter, which is set in the package ASCII configuration file. If `AUTO_RUN` is set to `YES`, then Package Switching is enabled initially when

the package first starts in the cluster. The initial setting for node switching is to allow switching to all nodes that are configured to run the package. Both node switching and package switching can be changed dynamically as the cluster is running.

### Changing Package Switching with Serviceguard Manager

To change package switching or node switching in Serviceguard Manager, select the package on the tree or map. Either right-click it or go to the toolbar Actions menu. Click on the radio button to change between enabled or disabled. You can set node switching node by node

Note that if the package is currently running on a node, and you disable switching of the package to that node, the package continues running, but it will not be able to switch back to the node at a later time.

To change the AUTO-RUN parameter, you need to reconfigure the package. This is only available for Serviceguard 11.16 or later. Select the package, then choose Configuring from the Actions menu (on toolbar or right-click menu). As on the command line, you need root permission on the cluster to create or modify configuration.

### Changing Package Switching with Serviceguard Commands

You can change package switching behavior either temporarily or permanently using Serviceguard commands. 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.

You can also disable package switching to particular nodes by using the `-n` option of the cmmodpkg command. The following disables the ability of pkg1 to switch to node lptest3:

```
# cmmodpkg -d -n lptest3 pkg1
```

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 subsequent section “Reconfiguring a Package on a Running Cluster” for detailed instructions on reconfiguration.

# Reconfiguring a Cluster

You can reconfigure a cluster either when it is halted or while it is still running. Some operations can only be done when the cluster is halted. Table 7-1 shows the required cluster state for many kinds of changes.

**Table 7-1 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. Serial heartbeat must not be configured. |
| Delete a node | A node can be deleted even though it is unavailable or unreachable. |
| Add a volume group | Cluster may be running or halted. |
| Delete a volume group | Cluster may be running or halted. Packages that use the volume group will not be able to start again until their control scripts are modified. |
| Change Maximum Configured Packages | Cluster must not be running. |
| Change Timing Parameters | Cluster must not be running. |
| Change Quorum Server Configuration | Cluster must not be running. |
| Change Cluster Lock Configuration | Cluster must not be running. |
| Change serial device files | Cluster must not be running. |
| Change IP addresses for heartbeats or monitored subnets | Cluster must not be running. |
| Change Access Control Policy (Serviceguard 11.16 or later) | Cluster and package may be running or halted. |

**Table 7-1 Types of Changes to Permanent Cluster Configuration**

| Change to the Cluster Configuration | Required Cluster State |
|---|---|
| Failover Optimization to enable or disable Faster Failover product | Cluster must not be running. |

## Reconfiguring a Halted Cluster

You can make a permanent change in cluster configuration when the cluster is halted. This procedure *must* be used for changes to the quorum server or lock disk 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, using Serviceguard's Halt Cluster command, or `cmhaltcl` on the command line.
2. On one node, reconfigure the cluster as described in the chapter "Building an HA Cluster Configuration." You can do this by using Serviceguard Manager (but only for Serviceguard version 11.16), or by issuing `cmquerycl` on the command line 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. To copy the binary cluster configuration file to all nodes, use Serviceguard's Apply button, or `cmapplyconf` on the command line. This file overwrites any previous version of the binary cluster configuration file.
4. Start the cluster on all nodes or on a subset of nodes. Use Serviceguard's Run Cluster Admin command, or `cmruncl` on the command line.

### Using Serviceguard Manager to Change MAX_CONFIGURED_PACKAGES

(Serviceguard 11.16 only, requires root login to the cluster.) First halt the cluster. Select it on the tree or map. From the Actions menu, select Administering Serviceguard, then Halt Cluster.

When halted, select the cluster in the map or tree. From the Actions menu, select Configuring Serviceguard. When the Configuring Cluster window opens, click the Parameters tab. Enter the new number. Click Apply. Close the configuration window. (After refresh, check the cluster's Properties to see the change.)

### Using Serviceguard Commands to Change MAX_CONFIGURED_PACKAGES

The cluster must be halted.

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. Time can be significantly reduced if you use the `-k` or `-K` options with `cmcheckconf`. Using `-k` or `-K` option with the `cmcheckconf` command, can significantly reduce the response time.

Use the `cmapplyconf` command to apply the changes to the configuration and send the new configuration file to all cluster nodes. Time can be significantly reduced if you use the `-k` or `-K` options with `cmapplyconf`.

## 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 change the quorum server or lock disk configuration while the cluster is running.
- You cannot remove an active node from the cluster. You must halt the node first.
- You cannot delete an active volume group from the cluster configuration. You must halt any package that uses the volume group and ensure that the volume is inactive before deleting it.
- 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).

Changes to the package configuration are described in a later section.

The following sections describe how to perform dynamic reconfiguration tasks using Serviceguard Manager or Serviceguard commands.

### Using Serviceguard Manager to Add Nodes to the Configuration While the Cluster is Running

Select the cluster on the tree or map. Choose Configuring from the Actions menu. You need root permission on the cluster. On the Nodes tab, under Available nodes, highlight the node you want to add, and click Add. Then click Apply. After Refresh, check the cluster's Properties to confirm the change.

### Using Serviceguard Commands to Add Nodes to the Configuration While the Cluster is Running

Use the following procedure to add a node with HP-UX commands. 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. Specify the node name (31 bytes or less) without its full domain name; for example, specify *ftsys8* and not *ftsys8.cup.hp.com*:

   ```
   # 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 `/etc/rc.config.d/cmcluster` file to enable the new node to join the cluster automatically each time it reboots.

**NOTE** If you add a node to a running cluster that uses CVM disk groups, the disk groups will be available for import when the node joins the cluster.

### Using Serviceguard Manager to Delete Nodes from the Configuration While the Cluster is Running

The node must be halted. If it is not, select it and choose Administering Serviceguard from the Actions menu. Choose Delete Node.

Select the cluster on the tree or map. Choose Configuring Serviceguard from the Actions menu. (You need root permission on the cluster.) On the Nodes tab, under Available nodes, highlight the node to remove, and click Delete. Then click Apply. After Refresh, check the cluster's Properties to confirm the change.

If the node you wish to delete is unreachable (disconnected from the LAN, for example), you can delete the node only if there are no packages which specify the unreachable node. If there are packages that depend on the unreachable node, halt the cluster or use Serviceguard commands as described in the next section.

### Using Serviceguard Commands to Delete Nodes from the Configuration While the Cluster is Running

Use the following procedure to delete a node with HP-UX commands. For this example, nodes *ftsys8*, *ftsys9* and *ftsys10* are already configured in a running cluster named *cluster1*, and you are deleting 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 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
   ```

3. Edit the file `clconfig.ascii` to check the information about the nodes that remain in the cluster.
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
   ```

**NOTE** If you are attempting to remove an unreachable node that has many packages dependent on it, especially if the dependent packages use a large number of EMS resources, you may see the following message:

```
The configuration change is too large to process while the
cluster is running.
Split the configuration change into multiple requests or halt
the cluster.
```

In this situation, you must halt the cluster to remove the node.

### Using Serviceguard Manager to Change the LVM Configuration While the Cluster is Running

Select the cluster on the tree or map. Choose Configuring Serviceguard from the Actions menu. (You need root permission on the cluster.) On the Logical Volumes tab highlight the node to add or remove, and click Add or Delete. Then click Apply. After Refresh, check the cluster's Properties to confirm the change.

You *cannot* change the cluster lock volume group or physical volume configuration while the cluster is running.

**NOTE** If you are removing a volume group from the cluster configuration, make sure that you also modify or delete any package control script that activates and deactivates this volume group. In addition, you should use the LVM `vgexport` command on the removed volume group from each node that will no longer be using the volume group.

### Using Serviceguard Commands to Change the LVM Configuration While the Cluster is Running

Use the `cmgetconf` command to obtain a current copy of the cluster's existing configuration. Example:

```
# cmgetconf -c clconfig.ascii
```

Edit the file `clconfig.ascii` to add or delete volume groups. 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.

Following is an example for adding two volume groups:

```
# cmcheckconf -C clconfig.ascii vgname1 vgname2
```

If there are a large number of volume groups, you can use a reference file that lists all the volume groups. Example, using a reference file:

```
# cmcheckconf -C clconfig.ascii vgreferencefile
```

Following are examples of removing the volume groups that were added in the preceding examples:

```
# cmcheckconf -C clconfig.ascii vgname1 -R vgname2
```

```
# cmcheckconf -C clconfig.ascii vgreferencefile
```

Use the cmapplyconf command to apply the changes to the configuration and send the new configuration file to all cluster nodes. Syntax for the options with `cmapplyconf` is the same as given for the cmcheckconf command above.

**NOTE** If you are deleting from the cluster a volume group that is currently activated by a package, the configuration will be changed but the deletion will not take effect until the package is halted; thereafter, the package will no longer be able to run without further modification, such as removing the volume group from the package control script.

### Changing the VxVM or CVM Storage Configuration

You can add VxVM disk groups to the cluster configuration while the cluster is running. To add new CVM disk groups, the cluster *must* be running.

Create CVM disk groups from this node. Open the configuration ASCII file of the package that uses the CVM storage; add the CVM storage group in a STORAGE_GROUP statement. Then issue the `cmapplyconf` command.

Similarly, you can delete VxVM or CVM disk groups provided they are not being used by a cluster node at the time.

**NOTE** If you are removing a disk group from the cluster configuration, make sure that you also modify or delete any package control script that imports and deports this disk group. If you are removing a CVM disk group, be sure to remove the `STORAGE_GROUP` entries for the disk group from the package ASCII file.

# Reconfiguring a Package

The process of reconfiguration of a package is somewhat like the basic configuration described in Chapter 6. Refer to that chapter for details on the configuration process.

The cluster can be either halted or running during package reconfiguration. The types of changes that can be made and the times when they take effect depend on whether the package is running or not.

## 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 this chapter. You can do this by using Serviceguard Manager or by editing the package ASCII file.
- Edit the package control script directly if not in Serviceguard's guided mode (in guided mode, Serviceguard manager does it automatically at reconfiguration). Any changes in service names will also require changes in the package configuration file.
- Click the Apply button in Serviceguard Manager, or enter `cmapplyconf` on the command line. This copies the binary cluster configuration file to all nodes. On the command line, 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. (Done automatically in Serviceguard Manager as part of Apply.)
- Use the Serviceguard Manager Run Cluster command, or enter `cmruncl` on the command line 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 in Serviceguard Manager, select it and then choose Configuring Serviceguard from the Actions menu. When the configuration window opens, choose options as described in Chapter 6. Alternatively, with HP-UX commands, use the following procedure (*pkg1* is used as an example):

1. Halt the package if necessary:

   ```
   # cmhaltpkg  pkg1
   ```

   See Table 7-2 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 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* in the cluster configuration file.

To create the package, follow the steps given in the chapter “Configuring Packages and Services.” If you are using the Serviceguard command line, however, *do not* specify the cluster ASCII file when verifying and

distributing the configuration with HP-UX commands. For example, to use HP-UX commands to verify the configuration of newly created *pkg1* on a running cluster:

```
# cmcheckconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

Use an HP-UX command like the following to distribute the new package configuration to all nodes in the cluster:

```
# cmapplyconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

Remember to copy the control script to the /etc/cmcluster/pkg1 directory on all nodes that can run the package.

## Deleting a Package from a Running Cluster

In Serviceguard Manager, first halt the package. Then, select the cluster. From the Actions menu, choose Configuring Serviceguard -> Delete Package.

On the Serviceguard command line, 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 `/etc/cmcluster/mypkg` is not deleted by this command.

## 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 7-2 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 7-2 Types of Changes to Packages**

| Change to the Package | Required Package State |
|---|---|
| Add a new package | Other packages may be in any state. |
| Delete a package | Package must not be running. |
| 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. |

**Table 7-2 Types of Changes to Packages (Continued)**

| Change to the Package | Required Package State |
|---|---|
| Remove a subnet | Package must not be running. |
| Add a resource | Package must not be running. |
| Remove a resource | Package must not be running. |
| Add a volume group | Volume group may be configured into the cluster 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. |
| 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 AutoRun | Package may be either running or halted. |
| Local LAN Failover | Package may be either running or halted. |

**Table 7-2 Types of Changes to Packages (Continued)**

| Change to the Package | Required Package State |
|---|---|
| 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. |
| Change access 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 Event Monitoring Service and its HA monitors can provide monitoring for disks, LAN cards, and some system events. Refer to the manual *Using HA Monitors* for more information.

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.

## Removing Serviceguard from a System

If you wish to remove a node from Serviceguard use, use the `swremove` 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, `swremove` 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 I/O Cards
- 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.

## Start the Cluster using Serviceguard Manager

If you have just finished configuring your cluster, it starts automatically. If it is halted later, restart it: select it the tree or map; from the Actions menu, choose Administering Serviceguard -> Run cluster.

## Testing the Package Manager

You can 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. Move the package back to the primary node using Serviceguard: Select the package. From the Actions menu, choose Administering Serviceguard -> Move Package.

## 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 SPU.
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 SPU.
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 original node using Serviceguard Manager.

   Select the package from the map or tree. From the Actions menu, choose Administering Packages -> Move package.
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, for each node on the cluster do the following:

1. To identify primary and standby lan cards on the node, enter

   ```
   # lanscan
   ```

   and then

   ```
   # cmviewcl -v
   ```

2. Disconnect the LAN connection from the Primary card. (Be careful not to break the subnet if you are using ThinLAN cables.)
3. Verify that a local switch has taken place so that the Standby card is now the Primary card. In Serviceguard Manager, check the cluster properties. Or, on the command line, use the `cmviewcl -v command`.
4. Reconnect the LAN to the original Primary card, and verify its status. In Serviceguard Manager, check the cluster properties. Or, on the command line, use the `cmviewcl -v` command.

# 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. The following should be monitored for errors or warnings of all kinds:

- Disks
- 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/adm/syslog/syslog.log) periodically for reports on all configured HA devices. The presence of errors relating to a device will show the need for maintenance.

## Using Event Monitoring Service

Event Monitoring Service (EMS) allows you to configure monitors of specific devices and system resources. You can direct alerts to an administrative workstation where operators can be notified of further action in case of a problem. For example, you could configure a disk monitor to report when a mirror was lost from a mirrored volume group being used in the cluster

Refer to the manual *Using HA Monitors* for additional information.

## Using EMS Hardware Monitors

A set of hardware monitors is available for monitoring and reporting on memory, CPU, and many other system values. Some of these monitors are supplied with specific hardware products.

Refer to the *EMS Hardware Monitors User's Guide* (B6191-90020) for additional information.

## Hardware Monitors and Persistence Requests

When hardware monitors are disabled using the `monconfig` tool, associated hardware monitor persistent requests are removed from the persistence files. When hardware monitoring is re-enabled, the monitor requests that were initialized using the `monconfig` tool are re-created.

However, hardware monitor requests created using Serviceguard Manager, or established when Serviceguard is started, are not re-created. These requests are related to the `psmmon` hardware monitor.

To re-create the persistence monitor requests, halt Serviceguard on the node, and then restart it. This will re-create the persistence monitor requests.

## Using HP Predictive Monitoring

In addition to messages reporting actual device failure, the logs may accumulate messages of lesser severity which, over time, can indicate that a failure may happen soon. One product that provides a degree of automation in monitoring is called HP Predictive, which gathers information from the status queues of a monitored system to see what errors are accumulating. This tool will report failures and will also predict failures based on statistics for devices that are experiencing specific non-fatal errors over time. In a Serviceguard cluster, HP Predictive should be run on all nodes.

HP Predictive also reports error conditions directly to an HP Response Center, alerting support personnel to the potential problem. HP Predictive is available through various support contracts. For more information, contact your HP representative.

# Replacing Disks

The procedure for replacing a faulty disk mechanism depends on the type of disk configuration you are using. Separate descriptions are provided for replacing an array mechanism and a disk in a high availability enclosure.

## Replacing a Faulty Array Mechanism

With any HA disk array configured in RAID 1 or RAID 5, refer to the array's documentation for instruction on how to replace a faulty mechanism. After the replacement, the device itself automatically rebuilds the missing data on the new disk. No LVM activity is needed. This process is known as **hot swapping** the disk.

## Replacing a Faulty Mechanism in an HA Enclosure

If you are using software mirroring with MirrorDisk/UX and the mirrored disks are mounted in a high availability disk enclosure, you can use the following steps to **hot plug** a disk mechanism:

1. Identify the physical volume name of the failed disk and the name of the volume group in which it was configured. In the following examples, the volume group name is shown as `/dev/vg_sg01` and the physical volume name is shown as `/dev/dsk/c2t3d0`. Substitute the volume group and physical volume names that are correct for your system.
2. Identify the names of any logical volumes that have extents defined on the failed physical volume.
3. On the node on which the volume group is currently activated, use the following command *for each logical volume that has extents on the failed physical volume*:

   ```
   # lvreduce -m 0 /dev/vg_sg01/lvolname /dev/dsk/c2t3d0
   ```

4. At this point, remove the failed disk and insert a new one. The new disk will have the same HP-UX device name as the old one.
5. On the node from which you issued the `lvreduce` command, issue the following command to restore the volume group configuration data to the newly inserted disk:

```
# vgcfgrestore -n /dev/vg_sg01 /dev/dsk/c2t3d0
```

6. Issue the following command to extend the logical volume to the newly inserted disk:

```
# lvextend -m 1 /dev/vg_sg01 /dev/dsk/c2t3d0
```

7. Finally, use the `lvsync` command *for each logical volume that has extents on the failed physical volume.* This synchronizes the extents of the new disk with the extents of the other mirror.

```
# lvsync /dev/vg_sg01/lvolname
```

## Replacing a Lock Disk

Replacing a failed lock disk mechanism is the same as replacing a data disk. If you are using a *dedicated* lock disk (one with no user data on it), then you need to issue only one LVM command, as in the following example:

```
# vgcfgrestore -n /dev/vg_lock /dev/dsk/c2t3d0
```

Serviceguard checks the lock disk on an hourly basis. After the `vgcfgrestore` command, review the syslog file of an active cluster node for not more than one hour. Then look for a message showing that the lock disk is healthy again.

## On-line Hardware Maintenance with In-line SCSI Terminator

In some shared SCSI bus configurations, on-line SCSI disk controller hardware repairs can be made if HP in-line terminator (ILT) cables are used. In-line terminator cables are supported with most SCSI-2 Fast-Wide configurations.

In-line terminator cables are supported with Ultra2 SCSI host bus adapters only when used with the SC10 disk enclosure. This is because the SC10 operates at slower SCSI bus speeds, which are safe for the use of ILT cables. In-line terminator cables are not supported for use in any Ultra160 or Ultra3 SCSI configuration, since the higher SCSI bus speeds can cause silent data corruption when the ILT cables are used.

The in-line terminator cable is available in a number of form factors; in each case, the connector at one end contains SCSI termination, which is used to terminate the end of the SCSI bus instead of the termination on the SCSI host bus adapter (HBA) in the node. If the terminated end of an

ILT cable is connected to an HBA, then termination must be disabled on that HBA. Disabling the termination is done on the HBA by removing the termination resistor packs, setting the appropriate DIP switches on the HBA, or by programmatically disabling termination, depending on the HBA being used. (Consult the documentation for the HBA to see which method works for a particular HBA.)

ILT cables can be used in combination with Y-cables to allow additional nodes to be connected to the shared SCSI bus. Some SCSI cables available from HP have combined ILT and Y-cable functionality. Any nodes connected to the middle connector of a Y-cable must also have SCSI termination disabled on the HBA's.

When an ILT cable is used, it is possible to physically disconnect a host from the end of the shared SCSI bus without breaking the bus's termination, allowing the remaining nodes in the cluster to continue to access the shared SCSI bus while the repairs are being made.

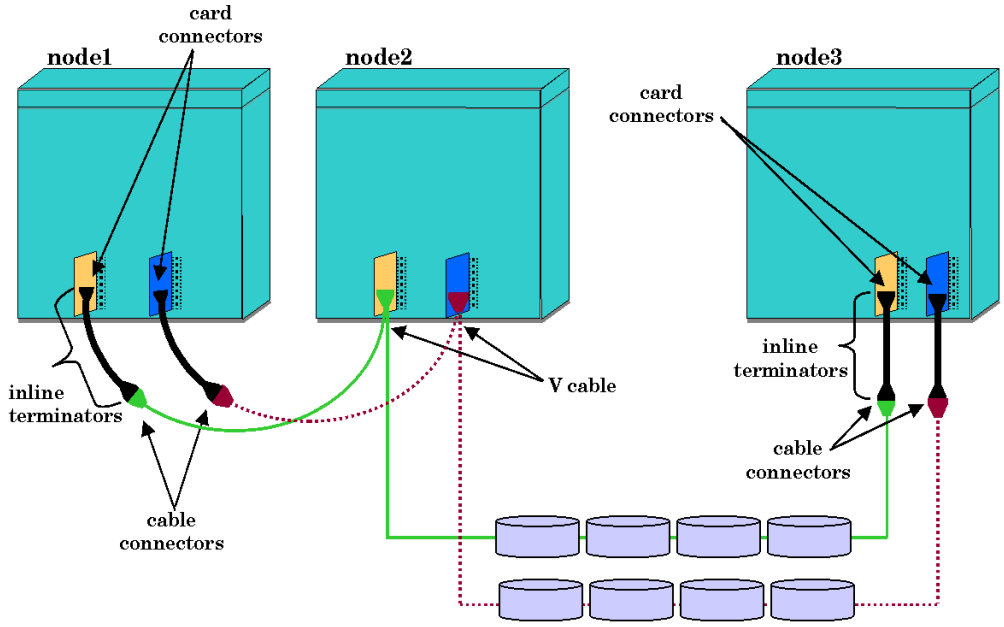
Similarly, it is possible to physically disconnect a host from the middle connector of a Y-cable on a shared SCSI bus without breaking the bus's termination.

Whether using ILT cables or Y-cables, however, it is strongly recommended that you do not try to physically reconnect an HBA to the shared SCSI bus without first halting all nodes connected to that shared SCSI bus. This is because it is very easy to inadvertently short out a pin in the connector against the chassis ground, which would damage the other HBA's connected to the shared SCSI bus, and bring the entire SCSI bus down.

**NOTE** You cannot use inline terminators with internal FW/SCSI buses on D and K series systems, and you cannot use the inline terminator with single-ended SCSI buses. You must *not* use an inline terminator to connect a node to a Y cable.

Figure 8-1 shows a three-node cluster with two F/W SCSI-2 buses. The solid line and the dotted line represent different buses, both of which have inline terminators attached to nodes 1 and 3. Y cables are also shown attached to node 2.

**Figure 8-1 F/W SCSI-2 Buses with In-line Terminators**




The use of in-line SCSI terminators allows you to do hardware maintenance on a given node by temporarily moving its packages to another node and then halting the original node while its hardware is serviced. Following the replacement, the packages can be moved back to the original node.

Use the following procedure to disconnect a node that is attached to the bus with an in-line SCSI terminator or with a Y cable:

1. Move any packages on the node that requires maintenance to a different node.
2. Halt the node that requires maintenance. The cluster will re-form, and activity will continue on other nodes. Packages on the halted node will switch to other available nodes if they are configured to switch.
3. Disconnect the power to the node.
4. Disconnect the node from the in-line terminator cable or Y cable if necessary. The other nodes accessing the bus will encounter no problems as long as the in-line terminator or Y cable remains connected to the bus.

5. Replace or upgrade hardware on the node, as needed.
6. Halt all nodes connected to the shared SCSI bus, and power them down.
7. Reconnect the node to the in-line terminator cable or Y cable if necessary.
8. Power-on the nodes in the cluster. If `AUTOSTART_CMCLD` is set to 1 in the `/etc/rc.config.d/cmcluster` file, the nodes will automatically start the cluster and the packages.
9. If necessary, move packages back to the node from their alternate locations and restart them.

## Replacement of I/O Cards

After an I/O card failure, you can replace the card using the following steps. It is not necessary to bring the cluster down to do this if you are using SCSI inline terminators or Y cables at each node.

1. Halt the node. In Serviceguard Manager, select the node; from the Actions menu, choose Administering Serviceguard -> Halt node. Or, from the Serviceguard command line, use the `cmhaltnode` command. Packages should fail over normally to other nodes.
2. Remove the I/O cable from the card. With SCSI inline terminators, this can be done without affecting the disks or other nodes on the bus.
3. Using SAM, select the option to do an on-line replacement of an I/O card.
4. Remove the defective I/O card.
5. Install the new 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.
6. In SAM, select the option to attach the new I/O card.
7. Add the node back into the cluster. In Serviceguard Manager, select the node; from the Actions menu, choose Administering Serviceguard -> Move node. Or, from the Serviceguard command line, issue the `cmrunnode` command.

# Replacement of LAN Cards

If you have a LAN card failure, which requires the LAN card to be replaced, you can replace it on-line or off-line depending on the type of hardware and operating system you are running. It is not necessary to bring the cluster down to do this.

## Off-Line Replacement

The following steps show how to replace a LAN card off-line. These steps apply to both HP-UX 11.0 and 11i:

1. Halt the node by using the `cmhaltnode` command.
2. Shut down the system using `/usr/sbin/shutdown`, then power down the system.
3. Remove the defective LAN card.
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.
5. Power up the system.
6. If necessary, add the node back into the cluster by using the `cmrunnode` command. (You can omit this step if the node is configured to join the cluster automatically.)

## On-Line Replacement

If your system hardware supports hotswap I/O cards, and if the system is running HP-UX 11i (B.11.11 or later), you have the option of replacing the defective LAN card on-line. This will significantly improve the overall availability of the system. To do this, follow the steps provided in the section “How to On-line Replace (OLR) a PCI Card Using SAM” in the document *Configuring HP-UX for Peripherals*. The OLR procedure also requires that the new card must be exactly the same card type as the card you removed to avoid improper operation of the network driver. Serviceguard will automatically recover the LAN card once it has been replaced and reconnected to the network.

## After Replacing the Card

After the on-line or off-line replacement of LAN cards has been done, 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.

It is also recommended that you update the new MAC address in the cluster binary configuration file by re-applying the cluster configuration. Use the following steps for on-line reconfiguration:

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 `cmviewconcl` and `lanscan` 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 (`/etc/cmcluster/qs_authfile`) 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.
4. Start the quorum server as follows:
   - Edit the `/etc/inittab` file to add the quorum server entries.
   - Use the `init q` command to run the quorum server.

   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 not 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.

**WARNING** **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 `netstat -in` 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 lan0 on *ftsys9* along with the heartbeat IP address.

```
Name      Mtu   Network         Address            Ipkts
   Opkts
ni0*      0     none            none                   0
       0
ni1*      0     none            none                   0
       0
lo0       4608  127             127.0.0.1          10114
   10114
lan0      1500  15.13.168       15.13.171.14      959269
  305189
```

```
lan0      1500  15.13.168       15.13.171.23      959269
  305189
lan0      1500  15.13.168       15.13.171.20      959269
  305189
lan1*     1500  none            none              418623
   41716
```

## 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 is `/var/adm/syslog/syslog.log`. 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 HP-UX in addition to Serviceguard use the syslog.log file to save messages. The HP-UX *Managing Systems and Workgroups* manual provides additional information on using the system log.

### Sample System Log Entries

The following entries from the file */var/adm/syslog/syslog.log* 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.

```
Dec 14 14:33:48 star04 cmcld[2048]: Starting cluster managemen
t protocols.
Dec 14 14:33:48 star04 cmcld[2048]: Attempting to form a new c
luster
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 mem
bership is:
    star04(id=1) , star05(id=2), star06(id=3)
```

```
Dec 14 17:33:53 star04 cmlvmd[2049]: Clvmd initialized success
fully.
Dec 14 14:34:44 star04 CM-CMD[2054]: cmrunpkg -v pkg5
Dec 14 14:34:44 star04 cmcld[2048]: Request from node star04 t
o start
    package pkg5 on node star04.
Dec 14 14:34:44 star04 cmcld[2048]: Executing '/etc/cmcluster/
pkg5/pkg5_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 ex
ited 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 managemen
t protocols.
Dec 14 14:39:27 star04 cmcld[2098]: Attempting to form a new c
luster
Dec 14 14:39:27 star04 cmclconfd[2097]: Command execution mess
age
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 mem
bership is:
   star04(id=1), star05(id=2), star06(id=3)
Dec 14 17:39:33 star04 cmlvmd[2099]: Clvmd initialized success
fully.
Dec 14 14:39:34 star04 cmcld[2098]: Executing '/etc/cmcluster/
pkg4/pkg4_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]: cmmodnet -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 no
de star04.
```

## Reviewing Object Manager Log Files

The Serviceguard Object Manager daemon `cmomd` logs messages to the file `/var/opt/cmom/cmomd.log`. You can review these messages using the `cmreadlog` command, as follows:

```
# cmreadlog /var/opt/cmom/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 Serviceguard Manager Log Files

Serviceguard Manager maintains a log file of user activity. This file is stored in the HP-UX directory `/var/opt/sgmgr` or the Windows directory `X:\Program Files\Hewlett-Packard\Serviceguard Manager\log` (where X refers to the drive on which you have installed Serviceguard Manager). You can review these messages using the `cmreadlog` command, as in the following HP-UX example:

```
# cmreadlog /var/opt/sgmgr/929917sgmgr.log
```

Messages from Serviceguard Manger include information about the login date and time, Object Manager server system, timestamp, etc.

## 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 at `/etc/cmcluster/`*`package_name`*`/`*`control_script`*`.log`. 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.

## Using the `cmcheckconf` Command

In addition, `cmcheckconf` can be used to troubleshoot your cluster just as it was used to verify the configuration.

The following example shows the commands used to verify the existing cluster configuration on *`ftsys9`* and *`ftsys10`*:

```
# cmquerycl -v -C /etc/cmcluster/verify.ascii -n ftsys9 -n
ftsys10
# cmcheckconf -v -C /etc/cmcluster/verify.ascii
```

The `cmcheckconf` command checks:

- The network addresses and connections.
- The cluster lock disk connectivity.
- 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.

## Using the `cmscancl` Command

The command `cmscancl` displays information about all the nodes in a cluster in a structured report that allows you to compare such items as IP addresses or subnets, physical volume names for disks, and other

node-specific items for all nodes in the cluster. `cmscancl` actually runs several different HP-UX commands on all nodes and gathers the output into a report on the node where you run the command.

The following are the types of configuration data that `cmscancl` displays for each node:

**Table 8-1 Data Displayed by the `cmscancl` Command**

| Description | Source of Data |
|---|---|
| LAN device configuration and status | `lanscan` command |
| network status and interfaces | `netstat` command |
| file systems | `mount` command |
| LVM configuration | `/etc/lvmtab` file |
| LVM physical volume group data | `/etc/lvmpvg` file |
| link level connectivity for all links | `linkloop` command |
| binary configuration file | `cmviewconf` command |

## Using the `cmviewconf` Command

`cmviewconf` allows you to examine the binary cluster configuration file, even when the cluster is not running. The command displays the content of this file on the node where you run the command.

## Reviewing the LAN Configuration

The following networking commands can be used to diagnose problems:

- `netstat -in` can be used to examine the LAN configuration. This command lists all IP addresses assigned to each LAN interface card.
- `lanscan` can also be used to examine the LAN configuration. This command lists the MAC addresses and status of all LAN interface cards on the node.
- `arp -a` can be used to check the arp tables.

- `landiag` is useful to display, diagnose, and reset LAN card information.
- `linkloop` verifies the communication between LAN cards at MAC address levels. For example, if you enter

  ```
  # linkloop -i4 0x08000993AB72
  ```

  you should see displayed the following message:

  ```
  Link Connectivity to LAN station: 0x08000993AB72  OK
  ```

- `cmscancl` can be used to verify that primary and standby LANs are on the same bridged net.
- `cmviewcl -v` shows the status of primary and standby 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.
- Problems with VxVM Disk Groups.
- Package Movement Errors.
- Node and Network Failures.
- Quorum Server Problems.

The first two categories of problems occur with the incorrect configuration of Serviceguard. The last category contains “normal” failures to which Serviceguard is designed to react and ensure the availability of your applications.

## 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 `nslookup` command on each cluster node to see whether name resolution is correct. For example:

```
# nslookup ftsys9

Name Server:  server1.cup.hp.com
Address:  15.13.168.63

Name:    ftsys9.cup.hp.com
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, or you can use a different LAN type.
- 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:

- `bdf` - to see if your package's volume group is mounted.
- `vgdisplay -v` - to see if all volumes are present.
- `lvdisplay -v` - to see if the mirrors are synchronized.
- `strings /etc/lvmtab` - to ensure that the configuration is correct.

- `ioscan -fnC disk` - to see physical disks.
- `diskinfo -v /dev/rdsk/cxtydz` - to display information about a disk.
- `lssf /dev/dsk/*d0` - to check LV and paths.
- `vxdg list` - to list VERITAS disk groups.
- `vxprint` - to show VERITAS disk group details.

### 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 `cmmodnet(1m)` command. First determine which package IP addresses are installed by inspecting the output resulting from running the command `netstat -in`. If any of the IP addresses specified in the package control script appear in the netstat output under the "Address" column for IPv4 or the "Address/Prefix" column for IPv6, use `cmmodnet` to remove them:

   ```
   # cmmodnet -r -i <ip-address> <subnet>
   ```

   where <*ip-address*> is the address in the "Address" or the "Address/Prefix column and <*subnet*> is the corresponding entry in the "Network" column for IPv4, or the prefix (which can be derived from the IPV6 address) for IPv6.

3. Ensure that package volume groups are deactivated. First unmount any package logical volumes which are being used for filesystems. This is determined by inspecting the output resulting from running the command `bdf -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.

   ```
   # cmmodpkg -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:

   ```
   # cmmodpkg -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. (If you are using Serviceguard Manager, set *Package Failfast* to *Enabled*.) 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.

## Problems with VxVM Disk Groups

This section describes some approaches to solving problems that may occur with VxVM disk groups in a cluster environment. For most problems, it is helpful to use the `vxdg list` command to display the disk groups currently imported on a specific node. Also, you should consult the package control script log files for messages associated with importing and deporting disk groups on particular nodes.

### Force Import and Deport After Node Failure

After certain failures, packages configured with VxVM disk groups will fail to start, and the following error will be seen in the package log file:

```
vxdg: Error dg_01 may still be imported on ftsys9
         ERROR:  Function check_dg failed
```

This can happen if a package is running on a node which then fails before the package control script can deport the disk group. In these cases, the host name of the node that had failed is still written on the disk group header.

When the package starts up on another node in the cluster, a series of messages is printed as in the following example (the hostname of the failed system is ftsys9, and the disk group is dg_01):

```
check_dg: Error dg_01 may still be imported on ftsys9

To correct this situation, logon to ftsys9 and
execute the following command:
vxdg deport dg_01
```

```
Once dg_01 has been deported from ftsys9,
this package may be restarted via either cmmodpkg(1M)
or cmrunpkg(1M).

In the event that ftsys9 is either powered off
or unable to boot, then dg_01 must be force
imported.

****************** WARNING**************************

The use of force import can lead to data corruption if
ftsys9 is still running and has dg_01
imported. It is imperative to positively determine that
ftsys9 is not running prior to performing the force
import. See -C option on vxdg(1M).

******************************************************

To force import dg_01, execute the following
commands on the local system:
    vxdg -tfC import $vg
    vxdg deport $vg
```

Follow the instructions in the message to use the force import option (-C) to allow the current node to import the disk group. Then deport the disk group, after which it can be used again by the package. Example:

```
# vxdg -tfC import dg_01
```

```
# vxdg deport dg_01
```

The force import will clear the host name currently written on the disks in the disk group, after which you can deport the disk group without error so it can then be imported by a package running on a different node.

**CAUTION** This force import procedure should only be used when you are certain the disk is not currently being accessed by another node. If you force import a disk that is already being accessed on another node, data corruption can result.

## 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:

- HPMC. This is a High Priority Machine Check, a system panic caused by a hardware error.
- TOC
- Panics
- Hangs
- Power failures

In the event of a TOC, a system dump is performed on the failed node and numerous messages are also displayed on the console.

You can use the following commands to check the status of your network and subnets:

- `netstat -in` - to display LAN status and check to see if the package IP is stacked on the LAN card.
- `lanscan` - to see if the LAN is on the primary interface or has switched to the standby interface.
- `arp -a` - to check the arp tables.
- `lanadmin` - to display, test, and reset the LAN cards.

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.

## Troubleshooting Quorum Server

### Authorization File Problems

The following kind of message in a Serviceguard node's syslog file or in the output of `cmviewcl -v` may indicate an authorization problem:

```
Access denied to quorum server 192.6.7.4
```

The reason may be that you have not updated the authorization file. Verify that the node is included in the file, and try using `/usr/lbin/qs -update` to re-read the authorization file.

### Timeout Problems

The following kinds of message in a Serviceguard node's syslog file may indicate timeout problems:

```
Unable to set client version at quorum server
192.6.7.2:reply timed out
Probe of quorum server 192.6.7.2 timed out
```

These messages could be an indication of an intermittent network; or the default quorum server timeout may not be sufficient. You can set the `QS_TIMEOUT_EXTENSION` to increase the timeout, or you can increase the heartbeat or node timeout value.

The following kind of message in a Serviceguard node's syslog file indicates that the node did not receive a reply to it's lock request on time. This could be because of delay in communication between the node and the qs or between the qs and other nodes in the cluster:

```
Attempt to get lock /sg/cluser1 unsuccessful. Reason:
request_timedout
```

### 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. The request will be retried a few seconds later and will succeed. The following message is logged:

```
Oct 008 16:10:06:0: Request for lock /sg/lockTest1
succeeded. New lock owners: 1,2.
```

# 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 MC/ServiceGuard Commands**

| Command | Description |
|---|---|
| `cmapplyconf` | Verify and apply ServiceGuard cluster configuration and package configuration files.<br><br>`cmapplyconf` verifies the cluster configuration and package configuration specified in the `cluster_ascii_file` and the associated `pkg_ascii_file`(s), creates or updates the binary configuration file, called `cmclconfig`, 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 `/etc/cmcluster` directory.<br><br>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 `cmcheckconf` command and then use cmapplyconf 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. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmapplyconf` (continued) | It is recommended that the user run the cmgetconf command to get either the cluster ASCII configuration file or package ASCII configuration file whenever changes to the existing configuration are required.<br><br>Note that `cmapplyconf` 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. |
| `cmcheckconf` | Check high availability cluster configuration and/or package configuration files.<br><br>`cmcheckconf` verifies the cluster configuration as specified by the `cluster_ascii_file` and/or the package configuration files specified by each `pkg_ascii_file` in the command. If the cluster has already been configured previously, the `cmcheckconf` command will compare the configuration in the `cluster_ascii_file` against the previously configuration information stored in the binary configuration file and validates the changes. The same rules apply to the `pkg_ascii_file`. It is necessary to halt the cluster to run the `cmcheckconf` command. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmdeleteconf` | Delete either the cluster or the package configuration.<br><br>`cmdeleteconf` deletes either the entire cluster configuration, including all its packages, or only the specified package configuration. If neither *`cluster_name`* nor *`package_name`* is specified, `cmdeleteconf` will delete the local cluster's configuration and all its packages. If only the *`package_name`* is specified, the configuration of *`package_name`* in the local cluster is deleted. If both *`cluster_name`* and *`package_name`* are specified, the package must be configured in the *`cluster_name`*, and only the package *`package_name`* will be deleted. The local cluster is the cluster that the node running the `cmdeleteconf` command belongs to. |
| `cmgetconf` | Get cluster or package configuration information.<br><br>`cmgetconf` obtains either the cluster configuration, not including the package configuration, or the specified package's configuration information, and writes to either the *`output_filename`* file, or to `stdout`. This command can be run whether the cluster is up or down. If neither *`cluster_name`* nor *`package_name`* is specified, `cmgetconf` will obtain the local cluster's configuration. If both *`cluster_name`* and *`package_name`* are specified, the package must be configured in the *`cluster_name`*, and only the package configuration for *`package_name`* will be written to *`output_filename`* or to `stdout`. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmhaltcl` | Halt a high availability cluster.<br>`cmhaltcl` causes all nodes in a configured cluster to stop their cluster daemons, optionally halting all packages or applications in the process.<br>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. |
| `cmhaltnode` | Halt a node in a high availability cluster.<br>`cmhaltnode` causes a node to halt its cluster daemon and remove itself from the existing cluster.<br>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.<br>If *`node_name`* is not specified, the cluster daemon running on the local node will be halted and removed from the existing cluster. |
| `cmhaltpkg` | Halt a high availability package.<br>`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. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmhaltserv` | 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.<br><br>`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.<br><br>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. |
| `cmmakepkg` | Create a high availability package template file.<br><br>`cmmakepkg` creates a template ASCII package configuration file or package control script as specified by the selected option. The *`output_file_name`* should be customized for a specific cluster environment. After customization, these files should be verified by the cmcheckconf command. If *`output_file_name`* is not provided, output will be directed to `stdout`. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmmodnet` | Add or remove an address from a high availability cluster.<br><br>`cmmodnet` is used in the high availability package control scripts to add or remove an *`IP_address`* from the current network interface running the given *`subnet_name`*.<br><br>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 relocatable 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. |
| `cmmodpkg` | Enable or disable switching attributes for a high availability package.<br><br>`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. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmquerycl` | Query cluster or node configuration information.<br><br>`cmquerycl` searches all specified nodes for cluster configuration and Logical Volume Manager (LVM) information. Cluster configuration information includes network information such as LAN interface, IP addresses, bridged networks and possible heartbeat networks. 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. |
| `cmreadlog` | Format an Object Manager log file for easy display.<br><br>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. |
| `cmruncl` | Run a high availability cluster.<br><br>`cmruncl` 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 cmrunnode command should be used to start the remaining nodes and force them to join the existing cluster. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmrunnode` | Run a node in a high availability cluster.<br>`cmrunnode` causes a node to start its cluster daemon to join the existing cluster<br>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. |
| `cmrunpkg` | Run a high availability package.<br>`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. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmrunserv` | 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.<br><br>`cmrunserv` is used in the high availability package run script to run a service. If the service process dies, `cmrunserv` 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.<br><br>Should the *`service_command`* be halted by the `cmhaltserv` 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. |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmscancl` | Gather system configuration information from nodes with ServiceGuard installed.<br><br>`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.<br><br>If output _file is not specified, the information will be directed to stdout. Output file contains:<br>• LAN device configuration (output from `lanscan`)<br>• network status and interfaces (output from `netstat`)<br>• file systems (output from `mount`)<br>• LVM configuration (contents of `/etc/lvm/tab` file)<br>• LVM physical vg information (contents of `/etc/lvmpvg` file)<br>• link-level connectivity (output from `linkloop`)<br>• binary configuration file data (output from `cmviewconf`) |

**Table A-1 MC/ServiceGuard Commands (Continued)**

| Command | Description |
|---|---|
| `cmstartres` | Starts resource monitoring on the local node for an EMS resource that is configured in a ServiceGuard package.<br><br>`cmstartres` starts resource monitoring for an EMS resource on the local node. This resource must be configured in the specified *`package_name`*. |
| `cmstopres` | Stops resource monitoring on the local node for an EMS resource that is configured in a ServiceGuard package.<br><br>`cmstopres` stops resource monitoring for an EMS resource on the local node. This resource must be configured in the specified *`package_name`*. |
| `cmviewcl` | View information about the current high availability cluster.<br><br>`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. |
| `cmviewconf` | View MC/ServiceGuard or ServiceGuard cluster configuration information.<br><br>`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. |

# B Enterprise Cluster Master Toolkit

The Enterprise Cluster Master Toolkit (ECMT) provides a group of example scripts and package configuration files for creating Serviceguard packages for several major database and internet software products. Each toolkit contains a README file that explains how to customize the package for your needs.

ECMT for HP-UX 11iv 1,

The ECMT can be installed on HP-UX 11i v1 (HP Product Number B5139EA)or 11i v2 (HP Product Number T1909BA).

ECMT includes toolkits from the following internet applications:

- HP Apache
- HP Tomcat
- HP CIFS/909i

ECMT includes toolkits from the following database applications:

- Oracle 9i
- Oracle10g
- Informix (11iv 1 only)
- Sybase (11iv 1 only)
- DB2 (11iv 1 only)
- Progress (11iv 1 only)

A separate NFS toolkit is available. Refer to *Managing Highly Available NFS* (HP Part Number B5140-90017) for more information.

Other application integration scripts are available from your HP representative.

# C 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 accidently 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 JFS
- 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.

## Use Raw Volumes

If your application uses data, use raw volumes rather than filesystems. Raw volumes do not require an `fsck` of the filesystem, thus eliminating one of the potentially lengthy steps during a failover.

## Evaluate the Use of JFS

If a file system must be used, a JFS offers significantly faster file system recovery as compared to an HFS. However, performance of the JFS may vary with the application.

## 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 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 availablity. 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 LANIC 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()`, 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(2)` 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(2)`, specifying the application name rather than the hostname. `gethostbyname(2)` will pass in the IP address of the application. This IP address will move with the application to the new node.

However, `gethostbyname(2)` 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.

To prevent one node from inadvertently accessing disks being used by the application on another node, HA software uses an exclusive access mechanism to enforce access by only one node at a time. This exclusive access applies to a volume group as a whole.

## 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.

For information about the supported Serviceguard releases for rolling upgrade, see the Serviceguard Release Notes for your version, at `http://docs.hp.com/hpux/ha`.

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.

# D 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 JFS filesystems 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.

# E Rolling Software Upgrades

You can upgrade the HP-UX operating system and the Serviceguard software one node at a time without bringing down your clusters. This process can also be used any time one system needs to be taken offline for hardware maintenance or patch installations. Until the process of upgrade is complete on all nodes, you cannot change the cluster configuration files, and you will not be able to use any of the features of the new Serviceguard release.

Rolling upgrade is supported with any of the supported revisions of Serviceguard. You can roll forward from any previous revision to any higher revision. For example, it is possible to roll from Serviceguard version A.10.05 on HP-UX 10.10 to version A.11.15 on HP-UX 11.11.

The sections in this appendix are as follows:

- Steps for Rolling Upgrades
- Example of Rolling Upgrade
- Limitations of Rolling Upgrades

## Steps for Rolling Upgrades

Use the following steps:

1. Halt the node you wish to upgrade. This will cause the node's packages to start up on an adoptive node. In Serviceguard Manager, select the node; from the Actions menu, choose Administering Serviceguard, Halt node. Or, on the Serviceguard command line, issue the `cmhaltnode` command.
2. Edit the `/etc/rc.config.d/cmcluster` file to include the following line:

   ```
   AUTOSTART_CMCLD = 0
   ```

3. Upgrade the node to the available HP-UX release, including Serviceguard. You can perform other software or hardware upgrades if you wish (such as installation of VERITAS Volume Manager software), provided you do not detach any SCSI cabling. Refer to the section on hardware maintenance in the “Troubleshooting” chapter.
4. Edit the /etc/rc.config.d/cmcluster file to include the following line:

   ```
   AUTOSTART_CMCLD = 1
   ```

5. Restart the cluster on the upgraded node. In Serviceguard Manager, select the node; from the Actions menu, choose Administering Serviceguard, Run node. Or, on the Serviceguard command line, issue the `cmrunnode` command.
6. Repeat this process for each node in the cluster.

**NOTE** Be sure to plan sufficient system capacity to allow moving the packages from node to node during the process without an unacceptable loss of performance.

If a cluster were to fail before the rolling upgrade was complete (perhaps due to a catastrophic power failure), the cluster can be restarted by entering the `cmruncl` command from a node which has been upgraded to the latest revision of the software.

## Keeping Kernels Consistent

If you change kernel parameters as a part of doing a rolling upgrade, be sure to change the parameters similarly on all nodes that can run the same packages in a failover scenario.

### Migrating `cmclnodelist` entries to A.11.16

The cmclnodelist file is deleted when you upgrade to Serviceguard Version A.11.16.

The information in it is migrated to the new Access Control Policy form. All the <*hostnode*> <*username*> pairs from the `cmclnodelist` file are now triplets in the cluster configuration file, and all have the role of Monitor. If you want to grant administration roles to non-root users, add more entries in the configuration file.

The cmclnodelist will remain until all nodes in the cluster are at Version A.11.16.

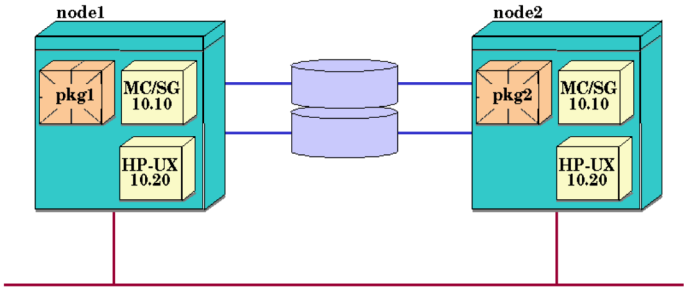
See “Editing Security Files” on page 182 for more information about the new access policies.

# Example of Rolling Upgrade

While you are performing a rolling upgrade warning messages may appear while the node is determining what version of software is running. This is a normal occurrence and not a cause for concern.

The following example shows a simple rolling upgrade on two nodes running one package each, as shown in Figure E-1. (This and the following figures show the starting point of the upgrade as Serviceguard 10.10 and HP-UX 10.20 for illustration only. A roll to Serviceguard 11.13 and HP-UX 11.00 is shown. For your systems, substitute the actual release numbers of your rolling upgrade path.)

**Figure E-1 Running Cluster Before Rolling Upgrade**



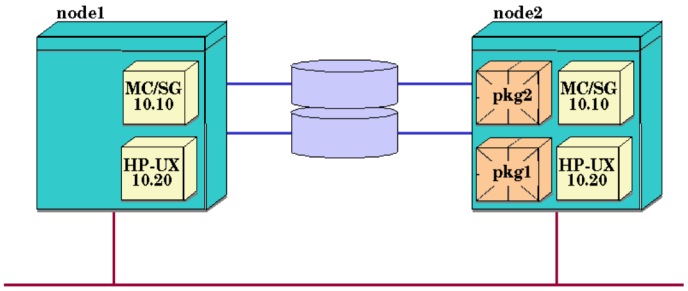

## Step 1.

Halt the first node, as follows

```
# cmhaltnode -f node1
```

This will cause PKG1 to be halted cleanly and moved to node 2. The Serviceguard daemon on node 1 is halted, and the result is shown in Figure E-2.

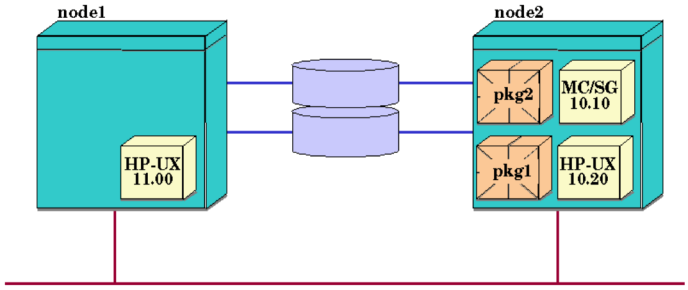
**Figure E-2 Running Cluster with Packages Moved to Node 2**




## Step 2.

Upgrade node 1 to the next operating system release (in this example, HP-UX 11.00), and install the next version of Serviceguard (11.13), as shown in Figure E-3.

**Figure E-3 Node 1 Upgraded to HP-UX 11.00**



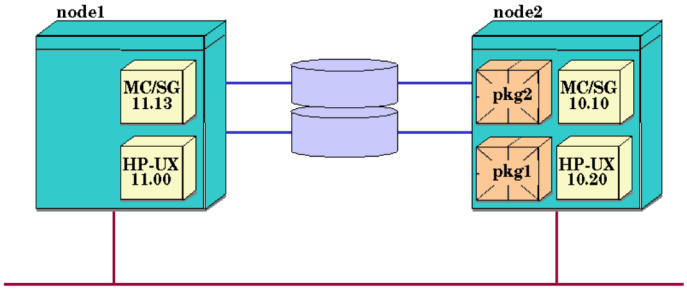

## Step 3.

When upgrading is finished, enter the following command on node 1 to restart the cluster on node 1.

```
# cmrunnode -n node1
```

At this point, different versions of the Serviceguard daemon (*cmcld*) are running on the two nodes, as shown in Figure E-4.

**Figure E-4 Node 1 Rejoining the Cluster**



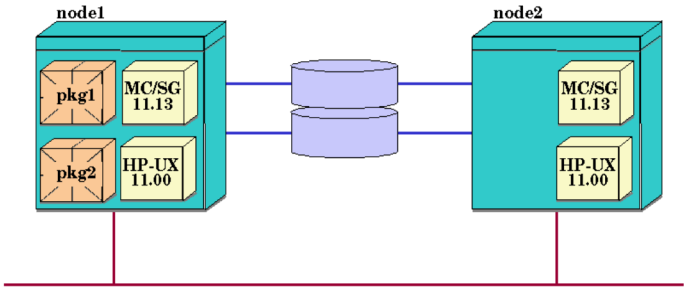

## Step 4.

Repeat the process on node 2. Halt the node, as follows:

```
# cmhaltnode -f node2
```

This causes both packages to move to node 1. Then upgrade node 2 to HP-UX 11.00 and Serviceguard 11.13.

**Figure E-5 Running Cluster with Packages Moved to Node 1**



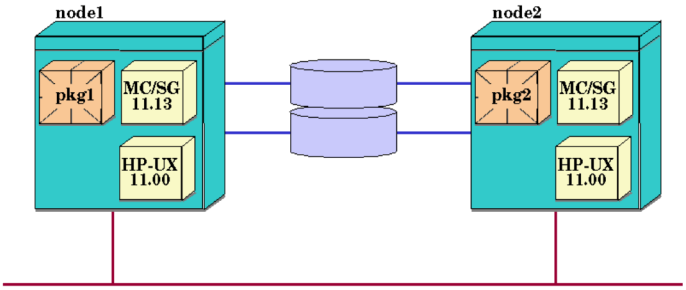

## Step 5.

Move PKG2 back to its original node. Use the following commands:

```
# cmhaltpkg pkg2
# cmrunpkg -n node2 pkg2
# cmmodpkg -e pkg2
```

The cmmodpkg command re-enables switching of the package, which is disabled by the cmhaltpkg command. The final running cluster is shown in Figure E-6.

**Figure E-6 Running Cluster After Upgrades**



## Limitations of Rolling Upgrades

The following limitations apply to rolling upgrades:

- During rolling upgrade, you should issue Serviceguard commands (other than `cmrunnode` and `cmhaltnode`) only on a node containing the latest revision of the software. Performing tasks on a node containing an earlier revision of the software will not work or will cause inconsistent results.
- You cannot modify the cluster or package configuration until the upgrade is complete. You *cannot* modify the hardware configuration—including the cluster's network configuration—during rolling upgrade. This means that you must upgrade all nodes to the new release before you can modify the configuration file and copy it to all nodes.
- None of the features of the newer release of Serviceguard are allowed until all nodes have been upgraded.
- Binary configuration files may be incompatible between releases of Serviceguard. Do *not* manually copy configuration files between nodes.
- Within a Serviceguard cluster, no more than two versions of Serviceguard can be running while the rolling upgrade is in progress.
- You can perform a rolling upgrade only on a configuration that has not been modified since the last time the cluster was started.
- Rolling upgrades are not intended as a means of using mixed releases of Serviceguard or HP-UX within the cluster. It is highly recommended that you upgrade all cluster nodes as quickly as possible to the new release level.
- You cannot delete Serviceguard software (via `swremove`) from a node while the cluster is in the process of rolling upgrade.

# F Blank Planning Worksheets

This appendix reprints blank versions of the planning worksheets described in the chapter “Planning and Documenting an HA Cluster.” You can duplicate any of these worksheets that you find useful and fill them in as a part of the planning process.

# Worksheet for Hardware Planning

```
                HARDWARE WORKSHEET                              Page ___ of ____

==============================================================================
    Node Information:

    Host Name _____________________     Series No _____________________

    Memory Capacity _____________________    Number of I/O Slots _____________

==============================================================================
    LAN Information:

    Name of           Name of             IP                Traffic
    Subnet _________  Interface __________ Addr_____________ Type   ___________

    Name of           Name of             IP                Traffic
    Subnet _________  Interface __________ Addr_____________ Type   ___________

    Name of           Name of             IP                Traffic
    Subnet _________  Interface __________ Addr_____________ Type   ___________


==============================================================================

   Serial Heartbeat Interface Information:

   Node Name _____________________  RS232 Device File _____________________

   Node Name _____________________  RS232 Device File _____________________


==============================================================================
    Disk I/O Information:

                      Hardware              Device
    Bus Type ______   Path    _____________ File Name  _____________

                      Hardware              Device
    Bus Type ______   Path    _____________ File Name  _____________

                      Hardware              Device
    Bus Type ______   Path    _____________ File Name  _____________
```

```
    Attach a printout of the output from ioscan -f and lssf /dev/*dsk/*s2
    after installing disk hardware and rebooting the system.  Mark this
    printout to indicate which physical volume group each disk belongs to.
.
```

# Power Supply Worksheet

```
                POWER SUPPLY WORKSHEET                          Page ___ of ____

================================================================================
    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: _____________________


================================================================================

    Quorum Services are Provided for:

Cluster Name: ___________________________________________________________

    Host Names ____________________________________________

    Host Names ____________________________________________

Cluster Name: ___________________________________________________________

    Host Names ____________________________________________

    Host Names ____________________________________________
```

# LVM Volume Group and Physical Volume Worksheet

```
PHYSICAL VOLUME WORKSHEET                    Page ___ of ____

===============================================================================

Volume Group Name: ____________________________________________________
                                PV Link 1              PV Link2

Physical Volume Name:_____________________________________________________

Physical Volume Name:_____________________________________________________

Physical Volume Name:_____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Volume Group Name: _____________________________________________________


                                PV Link 1              PV Link2

Physical Volume Name: ____________________________________________________

Physical Volume Name:_____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________

Physical Volume Name: ____________________________________________________
```

```
Physical Volume Name: _____________________________________________________

Physical Volume Name: _____________________________________________________

Physical Volume Name: _____________________________________________________
```

# VxVM Disk Group and Disk Worksheet

```
DISK GROUP WORKSHEET                    Page ___ of ____
================================================================
=============

Disk Group Name:
_____________________________________________________________

Physical Volume
Name:_____________________________________________________________

Physical Volume
Name:_____________________________________________________________

Physical Volume
Name:_____________________________________________________________

Physical Volume Name:
_____________________________________________________________

Physical Volume Name:
_____________________________________________________________

Physical Volume Name:
_____________________________________________________________

Physical Volume Name:
_____________________________________________________________

Physical Volume Name:
_____________________________________________________________


Disk Group Name:
_____________________________________________________________

Physical Volume Name:
_____________________________________________________________

Physical Volume
Name:_____________________________________________________________
```

```
Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________

Physical Volume Name:
_____________________________________________________
```

# Cluster Configuration Worksheet

```
=============================================================================
    Name and Nodes:
=============================================================================
    Cluster Name: _____________________________ OPS Version: _______________

    Node Names: ____________________________________________________________

    Volume Groups (for packages):_____________________________________________
    =========================================================================
    Subnets:

=============================================================================
    Heartbeat Subnet: _____________________________________

    Monitored Non-heartbeat Subnet: ______________________

    Monitored Non-heartbeat Subnet: ______________________
    =========================================================================
    Cluster Lock Volume Groups and Volumes:
=============================================================================
    First Lock Volume Group:     |        Physical Volume:
                                 |
        __________________       |   Name on Node 1: ____________________
                                 |
                                 |   Name on Node 2: ____________________
                                 |
                                 |   Disk Unit No: _________
                                 |
                                 |   Power Supply No: _________
    =========================================================================
    Timing Parameters:
=============================================================================
    Heartbeat Interval: ________
=============================================================================
    Node Timeout: _______
=============================================================================
    Network Polling Interval: ________
=============================================================================
.   Autostart Delay: ___________
=============================================================================
   Access Policies:
    User name:
```

```
    Host node:
    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?  _________ Local LAN Failover Allowed?  _________

    Node Failfast Enabled?      _________

    CVM Storage Groups:

    ___________________________________________________________________________

    ___________________________________________________________________________


    Additional Package Resource:

    Resource Name:___________ Polling Interval_______ Resource UP Value_________
================================================================================
   Access Policies:
    User name:
    Host node:
    Role:
================================================================================
```

# Package Control Script Worksheet

```
    LVM Volume Groups:

    VG[0]_____________VG[1]_____________VG[2]_____________

    VGCHANGE: _____________________________________________

    CVM Disk Groups:

    CVM_DG[0]_____________CVM_DG[1]_____________CVM_DG[2]_____________

    CVM_ACTIVATION_CMD: ___________________________________________

    VxVM Disk Groups:

    VXVM_DG[0]_____________VXVM_DG[1]_____________VXVM_DG[2]_____________

================================================================================
    Logical Volumes and File Systems:

    LV[0]_____________FS[0]_____________________________FS_MOUNT_OPT[0]_____

    LV[1]_____________________FS[1]_____________________FS_MOUNT_OPT[1]_____

    LV[2]_____________________FS[2]_____________________FS_MOUNT_OPT[2]_____

    FS Umount Count: _____________FS Mount Retry Count:_____________________
================================================================================
   Network Information:

IP[0] _____________________ SUBNET _____________________

    IP[1] _____________________ SUBNET _____________________
================================================================================
   Services:

    Service Name: _________  Command: _____________ Restart: _______

    Service Name: __________ Command: _____________ Restart: _______

================================================================================
```

```
Deferred Resources:

 Deferred Resource Name _____________________
```

# G Migrating from LVM to VxVM Data Storage

This appendix describes how to migrate LVM volume groups to VxVM disk groups for use with the VERITAS Volume Manager (VxVM) or with the Cluster Volume Manager (CVM). Topics are as follows:

- Loading VxVM
- Migrating Volume Groups
- Customizing Packages for VxVM
- Customizing Packages for CVM
- Removing LVM Volume Groups

The emphasis is on the steps you must take to manage the cluster and packages during migration; detailed instructions for configuring VxVM disk groups are given in the *VERITAS Volume Manager Administrator's Guide* and the *VERITAS Volume Manager Migration Guide*R at http://docs.hp.com/. Refer to Chapter 5 if you wish to create basic storage for a new system starting with fresh disks.

The procedures described below can be carried out while the cluster is running, but any package that uses a volume group that is being migrated must be halted. For disk groups that will be used with the Cluster Volume Manager (CVM), an additional set of steps is provided.

# Loading VxVM

Before you can begin migrating data, you must install the VERITAS Volume Manager software and all required VxVM licenses on all cluster nodes. This step requires each system to be rebooted, so it requires you to remove the node from the cluster before the installation, and restart the node after installation. This can be done as a part of a rolling upgrade procedure, described in Appendix E.

Details about VxVM installation are provided in the *VERITAS Volume Manager Release Notes*, available from http://www.docs.hp.com.

## Migrating Volume Groups

The following procedure shows how to do the migration of individual volume groups for packages that are configured to run on a given node. It is recommended to convert all the volume groups for a package at the same time.

It is assumed that VxVM software and an appropriate level of HP-UX and Serviceguard have been installed on the node, and that the node has rebooted and rejoined the cluster. It is further assumed that you have created a rootdg on the node as described above under "Creating a Root Disk Group."

1. Halt the package that activates the volume group you with to convert to VxVM:

   # **cmhaltpkg *PackageName***

2. Activate the LVM volume group in read-only mode:

   # **vgchange -a r *VolumeGroupName***

3. Back up the volume group's data, using whatever means are most appropriate for the data contained on this volume group. For example, you might use a backup/restore utility such as Omniback, or you might use an HP-UX utility such as `dd`.

4. Back up the volume group configuration:

   # **vgcfgbackup**

5. Define the new VxVM disk groups and logical volumes. You will need to have enough additional disks available to create a VxVM version of all LVM volume groups. You should create VxVM logical volumes that have the same general layout as the LVM configuration. For example, an LVM mirrored volume might have one mirror copy on one SCSI controller and a second copy on another controller to guard against a single controller failure disabling an entire volume. (Physical volume groups are sometimes used in LVM to enforce this separation.) The same mirroring pattern should be followed in creating the VxVM plexes, with different plexes configured on disks that are attached to different buses.

As an alternative to defining the VxVM disk groups on a new set of disks, it is possible to convert existing LVM volume groups into VxVM disk groups in line using the `vxvmconvert(1M)` utility. This utility is described along with its limitations and cautions in the *VERITAS Volume Manager Release Notes*, available from `http://www.docs.hp.com`. If using the `vxconvert(1M)` utility, then skip the next step and go ahead to the following section.

**NOTE**

Remember that the cluster lock disk must be configured on an LVM volume group and physical volume. If you have a lock volume group containing data that you wish to move to VxVM, you can do so, but do not use `vxvmconvert`, because the LVM header is still required for the lock disk.

6. Restore the data to the new VxVM disk groups. Use whatever means are most appropriate for the way in which the data was backed up in step 3 above.

## Customizing Packages for VxVM

After creating the VxVM disk group, you need to customize the Serviceguard package that will access the storage. Use the following procedure for disk groups that will be used with the VERITAS Volume Manager (VxVM). If you are using the Cluster Volume Manager (CVM), skip ahead to the next section.

1. Rename the old package control script as follows:

   # **mv *Package*.ctl *Package*.ctl.bak**

2. Create a new package control script with the same name as the old one:

   # **cmmakepkg -s *Package*.ctl**

3. Edit the new script to include the names of the new VxVM disk groups and logical volumes.

   The new portions of the package control script that are needed for VxVM use are as follows:

   - The `VXVM_DG[]` array. This defines the VxVM disk groups that are used for this package. The first `VxVM_DG[]` entry should be in index 0, the second in 1, etc. For example:

     ```
     VXVM_DG[0]="dg01"
     VXVM_DG[1]="dg02"
     ```

   - The `LV[]`, `FS[]` and `FS_MOUNT_OPT[]` arrays are used the same as they are for LVM. `LV[]` defines the logical volumes, `FS[]` defines the mount points, and `FS_MOUNT_OPT[]` defines any mount options. For example lets say we have two volumes defined in each of the two disk groups from above, lvol101, lvol102 and lvol201 and lvol202. These are mounted on /mnt_dg0101, /mnt_dg0102, /mnt_dg0201 and /mnt_dg0202 respectfully. /mnt_dg0101 and /mnt_dg0201 are both mounted read only. The `LV[]`, `FS[]` and `FS_MOUNT_OPT[]` entries for these would be as follows:

     ```
     LV[0]="/dev/vx/dsk/dg01/lvol101"
     LV[1]="/dev/vx/dsk/dg01/lvol0102"
     LV[2]="/dev/vx/dsk/dg02/lvol0201"
     LV[3]="/dev/vx/dsk/dg02/lvol0202"
     ```

```
FS[0]="/mnt_dg0101"
FS[1]="/mnt_dg0102"
FS[2]="/mnt_dg0201"
FS[3]="/mnt_dg0202"

FS_MOUNT_OPT[0]="-o ro"
FS_MOUNT_OPT[1]="-o rw"
FS_MOUNT_OPT[2]="-o ro"
FS_MOUNT_OPT[3]="-o rw"
```

4. Be sure to copy from the old script any user-specific code that may have been added, including environment variables and customer defined functions.
5. Distribute the new package control scripts to all nodes in the cluster.
6. Test to make sure the disk group and data are intact.
7. Deport the disk group:

   # **vxdg deport *DiskGroupName***

8. Make the disk group visible to the other nodes in the cluster by issuing the following command on all other nodes:

   # **vxdctl enable**

9. Restart the package.

## Customizing Packages for CVM

After creating the VxVM disk group, you need to customize the Serviceguard package that will access the storage. Use the following procedure if you will be using the disk groups with the Cluster Volume Manager (CVM). If you are using the VERITAS Volume Manager (VxVM), use the procedure in the previous section.

1. Rename the old package control script as follows:

   # **mv *Package*.ctl *Package*.ctl.bak**

2. Create a new package control script with the same name as the old one:

   # **cmmakepkg -s *Package*.ctl**

3. Edit the new script to include the names of the new CVM disk groups and logical volumes.

   The new portions of the package control script that are needed for CVM use are as follows:

   - The `CVM_DG[]` array. This defines the CVM disk groups that are used for this package. The first `CVM_DG[]` entry should be in index 0, the second in 1, etc. For example:

     ```
     CVM_DG[0]="dg01"
     CVM_DG[1]="dg02"
     ```

   - The `LV[]`, `FS[]` and `FS_MOUNT_OPT[]` arrays are used the same as they are for LVM. `LV[]` defines the logical volumes, `FS[]` defines the mount points, and `FS_MOUNT_OPT[]` defines any mount options. For example lets say we have two volumes defined in each of the two disk groups from above, lvol101, lvol102 and lvol201 and lvol202. These are mounted on /mnt_dg0101, /mnt_dg0102, /mnt_dg0201 and /mnt_dg0202 respectfully. /mnt_dg0101 and /mnt_dg0201 are both mounted read only. The `LV[]`, `FS[]` and `FS_MOUNT_OPT[]` entries for these would be as follows:

     ```
     LV[0]="/dev/vx/dsk/dg01/lvol101"
     LV[1]="/dev/vx/dsk/dg01/lvol0102"
     LV[2]="/dev/vx/dsk/dg02/lvol0201"
     LV[3]="/dev/vx/dsk/dg02/lvol0202"
     ```

```
FS[0]="/mnt_dg0101"
FS[1]="/mnt_dg0102"
FS[2]="/mnt_dg0201"
FS[3]="/mnt_dg0202"

FS_MOUNT_OPT[0]="-o ro"
FS_MOUNT_OPT[1]="-o rw"
FS_MOUNT_OPT[2]="-o ro"
FS_MOUNT_OPT[3]="-o rw"
```

4. Be sure to copy from the old script any user-specific code that may have been added, including environment variables and customer defined functions.
5. Be sure to uncomment the appropriate CVM_ACTIVATION_CMD statement to specify the kind of import you wish the package to perform on the disk group.
6. Distribute the new package control scripts to all nodes in the cluster.
7. Enter each disk group into the package ASCII configuration file immediately following the HALT_SCRIPT_TIMEOUT parameter. Add one STORAGE_GROUP definition for each disk group. For the two disk groups in the previous example, you would enter the following lines:

   ```
   STORAGE_GROUP dg01
   STORAGE_GROUP dg02
   ```

   Then re-apply the package configuration:

   # **cmapplyconf -P *PackageName*.ascii**
8. Test to make sure the disk group and data are intact.
9. Deport the disk group:

   # **vxdg deport *DiskGroupName***
10. Start the cluster, if it is not already running:

    # **cmruncl**

    This will activate the special CVM package.

11. When CVM starts up, it selects a master node, and this is the node from which you must issue the disk group configuration commands. To determine the master node, issue the following command from each node in the cluster:

    # **vxdctl -c mode**

    One node will identify itself as the master.

12. Make the disk group visible to the other nodes in the cluster by issuing the following command on the master node:

**# vxdg -s import *DiskGroupName***

13. Restart the package.

## Removing LVM Volume Groups

After testing the new VxVM disk groups, remove any LVM volume groups that are no longer wanted from the system using the standard LVM commands `lvremove`, `pvremove`, and `vgremove`. At a convenient time, you should also edit the cluster ASCII configuration file to remove the `VOLUME_GROUP` statements that refer to the LVM volume groups that are no longer used in the cluster. These entries should be removed before the next time you re-apply the cluster configuration.

# H IPv6 Network Support

This appendix describes some of the characteristics of IPv6 network addresses. Topics:

- IPv6 Address Types
- Network Configuration Restrictions
- Local Primary/Standby LAN Patterns
- Duplicate Address Detection Feature

# IPv6 Address Types

Several IPv6 types of addressing schemes are specified in the RFC 2373 (IPv6 Addressing Architecture). IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces. There are various address formats for IPv6 defined by the RFC 2373. IPv6 addresses are broadly classified as follows:

The following table explains the three types of IPv6 address types: unicast, anycast, and multicast.

**Table H-1 IPv6 Address Types**

| | |
|---|---|
| **Unicast** | An address for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. |
| **Anycast** | An address for a set of interfaces. In most cases these interfaces belong to different nodes. A packet sent to an anycast address is delivered to one of these interfaces identified by the address. Since the standards for using anycast addresses is still evolving, they are not supported in HP-UX as of now. |
| **Multicast** | An address for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address will be delivered to all interfaces identified by that address. |

Unlike IPv4, there are no broadcast addresses in IPv6 because their functions are superseded by multicast.

## Textual Representation of IPv6 Addresses

There are three conventional forms for representing IPv6 addresses as text strings:

- The first form is x:x:x:x:x:x:x:x, where 'x's are the hexadecimal values of eight 16-bit pieces of the 128-bit address. Example:

  2001:fecd:ba23:cd1f:dcb1:1010:9234:4088.

- Some of the IPv6 addresses may contain a long strings of zero bits. In order to make it easy for representing such addresses textually a special syntax is available. The use of "::" indicates that there are

multiple groups of 16-bits of zeros. The "::" can appear only once in an address and it can be used to compress the leading, trailing, or contiguous sixteen-bit zeroes in an address. Example:

fec0:1:0:0:0:0:0:1234 can be represented as fec0:1::1234.

- When dealing with a mixed environment of IPv4 and IPv6 nodes there is an alternative form of IPv6 address that will be used. It is x:x:x:x:x:x:d.d.d.d, where 'x's are the hexadecimal values of higher order 96 bits of IPv6 address and the 'd's are the decimal values of the 32-bit lower order bits. Typically IPv4 Mapped IPv6 addresses and IPv4 Compatible IPv6 addresses will be represented in this notation. These addresses will be discussed in later sections.

  Examples:

  0:0:0:0:0:0:10.1.2.3

  and

  ::10.11.3.123

## IPv6 Address Prefix

IPv6 Address Prefix is similar to CIDR in IPv4 and is written in CIDR notation. An IPv6 address prefix is represented by the notation:

IPv6-address/prefix-length

where "ipv6-address" is an IPv6 address in any notation listed above and "prefix-length" is a decimal value representing how many of the left most contiguous bits of the address comprise the prefix. Example:

fec0:0:0:1::1234/64.

The first 64-bits of the address "fec0:0:0:1" forms the address prefix. An address prefix is used in IPv6 addresses to denote how many bits in the IPv6 address represent the subnet.

## Unicast Addresses

IPv6 unicast addresses are classified into different types. They are global aggregatable unicast address, site-local address and link-local address. Typically a unicast address is logically divided as follows:

**Table H-2**

| n bits | 128-n bits |
|---|---|
| Subnet prefix | Interface ID |

Interface identifiers in a IPv6 unicast address are used to identify the interfaces on a link. Interface identifiers are required to be unique on that link. The link is generally identified by the subnet prefix.

A unicast address is called an unspecified address if all the bits in the address are zero. Textually it is represented as "::".

The unicast address "::1" or "0:0:0:0:0:0:0:1" is called the loopback address. It is used by a node to send packets to itself.

## IPv4 and IPv6 Compatibility

There are a number of techniques for using IPv4 addresses within the framework of IPv6 addressing.

### IPv4 Compatible IPv6 Addresses

The IPv6 transition mechanisms use a technique for tunneling IPv6 packets over the existing IPv4 infrastructure. IPv6 nodes that support such mechanisms use a special kind of IPv6 addresses that carry IPv4 addresses in their lower order 32-bits. These addresses are called IPv4 Compatible IPv6 addresses. They are represented as follows:

**Table H-3**

| 80 bits | 16 bits | 32 bits |
|---|---|---|
| zeros | 0000 | IPv4 address |

Example:

::192.168.0.1

### IPv4 Mapped IPv6 Address

There is a special type of IPv6 address that holds an embedded IPv4 address. This address is used to represent the addresses of IPv4-only nodes as IPv6 addresses. These addresses are used especially by applications that support both IPv6 and IPv4. These addresses are called as IPv4 Mapped IPv6 Addresses. The format of these address is as follows:

**Table H-4**

| 80 bits | 16 bits | 32 bits |
|---|---|---|
| zeros | FFFF | IPv4 address |

Example:

::ffff:192.168.0.1

### Aggregatable Global Unicast Addresses

The global unicast addresses are globally unique IPv6 addresses. This address format is very well defined in the RFC 2374 (An IPv6 Aggregatable Global Unicast Address Format). The format is:

**Table H-5**

| 3 | 13 | 8 | 24 | 16 | 64 bits |
|---|---|---|---|---|---|
| FP | TLA ID | RES | NLA ID | SLA ID | Interface ID |

where

FP = Format prefix. Value of this is "001" for Aggregatable Global unicast addresses.

TLA ID = Top-level Aggregation Identifier.

RES = Reserved for future use.

NLA ID = Next-Level Aggregation Identifier.

SLA ID = Site-Level Aggregation Identifier.

Interface ID = Interface Identifier.

### Link-Local Addresses

Link-local addresses have the following format:

**Table H-6**

| 10 bits | 54 bits | 64 bits |
|---|---|---|
| 1111111010 | 0 | interface ID |

Link-local address are supposed to be used for addressing nodes on a single link. Packets originating from or destined to a link-local address will not be forwarded by a router.

### Site-Local Addresses

Site-local addresses have the following format:

**Table H-7**

| 10 bits | 38 bits | 16 bits | 64 bits |
|---|---|---|---|
| 1111111011 | 0 | subnet ID | interface ID |

Link-local address are supposed to be used within a site. Routers will not forward any packet with site-local source or destination address outside the site.

### Multicast Addresses

A multicast address is an identifier for a group of nodes. Multicast addresses have the following format:

**Table H-8**

| 8 bits | 4 bits | 4 bits | 112 bits |
|---|---|---|---|
| 11111111 | flags | scop | group ID |

“FF” at the beginning of the address identifies the address as a multicast address.

The “flgs” field is a set of 4 flags “000T”. The higher order 3 bits are reserved and must be zero. The last bit ‘T’ indicates whether it is permanently assigned or not. A value of zero indicates that it is permanently assigned otherwise it is a temporary assignment.

The “scop” field is a 4-bit field which is used to limit the scope of the multicast group. For example, a value of ‘1’ indicates that it is a node-local multicast group. A value of ‘2’ indicates that the scope is link-local.

The “group ID” field identifies the multicast group. Some frequently used multicast groups are the following:

All Node Addresses = FF02:0:0:0:0:0:0:1 (link-local)

All Router Addresses = FF02:0:0:0:0:0:0:2 (link-local)

All Router Addresses = FF05:0:0:0:0:0:0:2 (site-local)

# Network Configuration Restrictions

Serviceguard now supports IPv6 for data links only. The heartbeat IP must still be IPv4, but the package IPs can be IPv4 or IPv6.

To configure IPv6, the system should be set up in what is called a dual-stack configuration which requires the IPv6 product bundle (IPv6NCF11i B.11.11.0109.5C) installed. Additional patches required after the installation of this bundle are listed later in this document, “Required and Recommended Patches.”

The restrictions for supporting IPv6 in Serviceguard are listed below.

- The heartbeat IP address must be IPv4. Therefore, IPv6-only operation nodes or IPv6-only nodes are not supported in a Serviceguard environment.
- The hostnames in a Serviceguard configuration must be IPv4. Serviceguard does not recognize IPv6 hostnames.
- Auto-configured IPv6 addresses are *not* supported in Serviceguard. as STATIONARY_IP addresses. All IPv6 addresses that are part of a Serviceguard cluster configuration must *not* be auto-configured through router advertisements, for example. They must be manually configured in `/etc/rc.config.d/netconf-ipv6`.
- Link-local IP addresses are *not* supported, either as package IPs or as STATIONARY_IPs. Depending on the requirements, the package IP could be of type site-local or global.
- Serviceguard supports only one IPv6 address belonging to each scope type (site-local and global) on each network interface (that is, restricted multi-netting). Therefore, up to a maximum of two IPv6 STATIONARY_IPS can be mentioned in the cluster ascii file for a NETWORK_INTERFACE: one being the site-local IPv6 address, and the other being the global IPv6 address.
- Quorum server, if used, has to be configured on an IPv4 network. It is not IPv6-capable. A quorum server configured on an IPv4 network can still be used by Serviceguard IPv6 clusters that have IPv6 networks as a part of their cluster configuration.
- Serviceguard supports IPv6 only on the Ethernet networks, including 10BT, 100BT, and Gigabit Ethernet

**NOTE** Even though link-local IP addresses are not supported in the Serviceguard cluster configuration, the primary link-local address on the Serviceguard primary interface will be switched over the standby during a local switch. This is because of two requirements: First, the dual stack (IPv4/IPv6) kernel requires that the primary IP address associated with an interface must always be a link-local address. Second, Serviceguard requires that the site-local and global IPs be switched to the standby network interface.

## IPv6 Relocatable Address and Duplicate Address Detection Feature

The IPv6 networking stack has a new feature, Duplicate Address Detection (DAD), that was not previously available in IPv4. When an address is being added, the DAD detects a duplicate address that is already being used on the network. It sends out a multicast message to the network neighborhood, and requires at least one second to listen for responses from other nodes. If no responses are received in that time, the relocatable IPv6 address is considered free to use. For more information regarding this feature, please refer to the RFC 2462.

The effect of this feature on Serviceguard is that the time required to add each IPv6 relocatable address will be at least one second longer than adding a corresponding IPv4 address. Depending on the number of IPv6 addresses configured within a package, this could have a moderate to significant impact on package start time.

If you do not need duplicate address detection, you can disable the DAD feature by setting the kernel parameter ip6_nd_dad_solicity_count to 0. Please note that this kernel parameter applies to the entire system. If you turn it off, you disable it for all applications on the system. For systems where DAD is not required, disabling this feature can significantly improve the start time of package packages containing a large number of IPv6 relocatable addresses.

To determine the current state of DAD on your system, use the `ndd -get` command to see the current value of the kernel parameter.

```
# ndd -get /dev/ip6 ip6_nd_dad_solicit_count
```

If the result is 1, the feature is turned on. If the result is 0, the feature is turned off.

To temporarily change the state of DAD on your computer, use the ndd -set command to change the kernel parameter.

```
# ndd -set /dev/ip6 ip6_nd_dad_solicit_count n
```

where *n* is a number: either 1 to turn the feature on, or 0 to turn it off.

To change the state of DAD on your computer so that it will remain changed even after a reboot, add the following entries to the /etc/rc/config.d/nddconf file:

```
# TRANSPORT_NAME[index]=ip6

# NDD_NAME[index]=ip6_nd_dad_solicit_count

# NDD_VALUE[index]=n
```

Where *index* is the next available integer value of the nddconf file, and *n* is a number: either 1 to turn the feature ON or 0 to turn it OFF.

# Local Primary/Standby LAN Patterns

The use of IPv6 allows a number of different patterns of failover among LAN cards configured in the cluster. This is true because each LAN card can support several IP addresses when a dual IPv4/IPv6 configuration is used. This section describes several ways in that local failover to a standby LAN can be configured.

By definition, a standby network interface is an interface that has no IP address(es) of either address family (IPv4 or IPv6) and is bridged to the primary network interface on a node.

Here are two guidelines to keep in mind when using IPv4 and IPv6 addresses in local failover situations:

- Since a network interface card can have both an IPv4 and an IPv6 address as the primary IPs, the standby network interface card could potentially act as a standby for both types of primary interfaces.

  However, if the IPv4 and IPv6 address(es) are configured on two separate network interfaces, then the standby interface can take over the IP address from only one network interface during a local failover.

  That is, IPv4 and IPv6 addresses from two separate network interfaces are mutually exclusive in a failover condition.

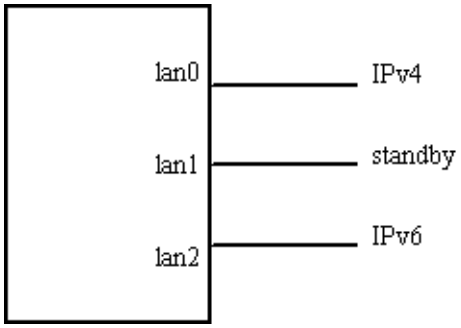
- Serviceguard will switch over link-local address configured on the primary network interface along with all other IP addresses which are configured as part of the cluster configuration to the standby network interface. This includes all heartbeat and stationary IPs (IPv4 and IPv6) and package IPs (both IPv4 and IPv6) added by Serviceguard.

The examples that follow illustrate this.

# Example Configurations

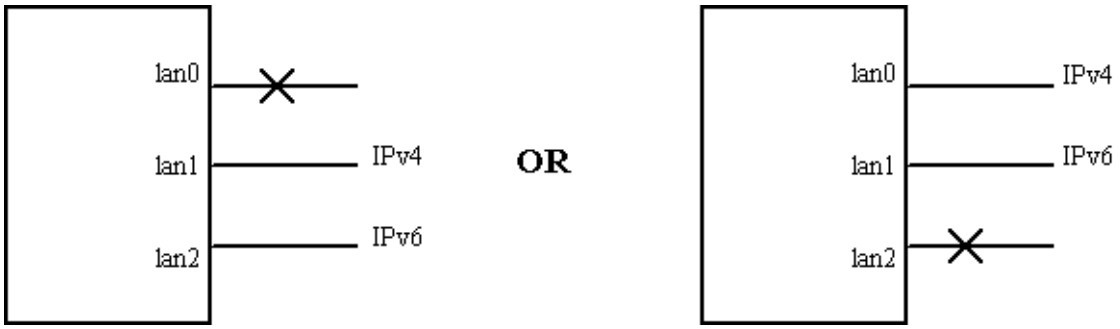
An example of a LAN configuration on a cluster node using both IPv4 and IPv6 addresses is shown in below.

**Figure H-1 Example 1: IPv4 and IPv6 Addresses in Standby Configuration**



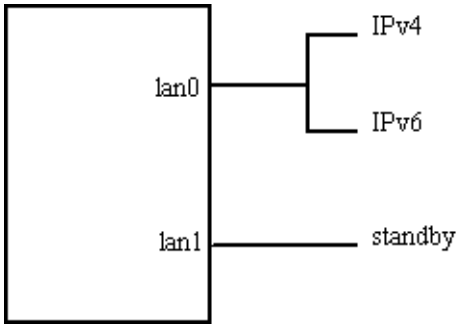

Following the loss of lan0 or lan2, lan1 can adopt either address, as shown below.

**Figure H-2 Example 1: IPv4 and IPv6 Addresses after Failover to Standby**



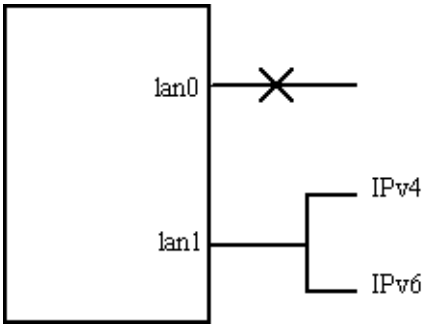

The same LAN card can be configured with both IPv4 and IPv6 addresses, as shown in below.

**Figure H-3 Example 2: IPv4 and IPv6 Addresses in Standby Configuration**




This type of configuration allows failover of both addresses to the standby. This is shown in below.

**Figure H-4 Example 2: IPv4 and IPv6 Addresses After Failover to Standby**



## Duplicate Address Detection Feature

The IPv6 networking stack has a new feature, Duplicate Address Detection (DAD), that was not previously available in IPv4. When an address is being added, the DAD detects a duplicate address that is already being used on the network. It sends out a multicast message to the network neighborhood, and requires at least one second to listen for responses from other nodes. If no responses are received in that time, the relocatable IPv6 address is considered free to use. For more information regarding this feature, please refer to the RFC 2462.

The effect of this feature on Serviceguard is that the time required to add each IPv6 relocatable address will be at least one second longer than adding a corresponding IPv4 address. Depending on the number of IPv6 addresses configured within a package, this could have a moderate to significant impact on package start time.

If you do not need duplicate address detection, you can disable the DAD feature by setting the kernel parameter ip6_nd_dad_solicity_count to 0. Please note that this kernel parameter applies to the entire system. If you turn it off, you disable it for all applications on the system. For systems where DAD is not required, disabling this feature can significantly improve the start time of package packages containing a large number of IPv6 relocatable addresses.

To determine the current state of DAD on your system, use the `ndd -get` command to see the current value of the kernel parameter.

```
# ndd -get /dev/ip6 ip6_nd_dad_solicit_count
```

If the result is 1, the feature is turned on. If the result is 0, the feature is turned off.

To change the state of DAD on your system, use the ndd -set command to change the kernel parameter.

```
# ndd -set /dev/ip6 ip6_nd_dad_solicit_count n
```

where *n* is a number: either 1 to turn the feature on, or 0 to turn it off.

To permanently change the state of DAD on your computer, add the following entries to the `/etc/rc/config.d/nddconf` file:

```
# TRANSPORT_NAME[index]=ip6

# NDD_NAME[index]=ip6_nd_dad_solicit_count
```

```
# NDD_VALUE[index]=n
```

Where *index* is the next available integer value of the `nddconf` file, and *n* is a number: either 1 to turn the feature ON or 0 to turn it OFF.

**D**

**O**



**P**

**S**

**T**



**U**



**V**

**W**