Managing Serviceguard Nineteenth Edition

HP Part Number: 5900-1492
Published: June 2011

Legal Notices

Contents

Publishing History

Table 1 Publishing History

Publishing 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 2005	B3936-90076	Eleventh, First reprint
October 2005	B3936-90095	Twelfth
December 2005	B3936-90095	Twelfth, First Reprint
March 2006	B3936-90100	Twelfth, Second Reprint
February 2007	B3936-90105	Thirteenth
June 2007	B3936-90117	Fourteenth
December 2007	B3936-90122	Fifteenth
May 2008	B3936-90135	Fifteenth, First Reprint
March 2009	B3936-90140	Sixteenth
July 2009	B3936-90143	Seventeenth
December 2009	B3936-90144	Seventeenth, First Reprint
September 2010	B3936-90147	Eighteenth
April 2011	5900–1492	Nineteenth
June 2011	5900–1492	Nineteenth, First Reprint

The last publishing date and part number indicate the current edition, which applies to Serviceguard version A.11.20 for the April 2011 release. See the latest edition of the Release Notes for a summary of changes in that release.

Preface

This nineteenth edition of the manual applies to Serviceguard Version A.11.20 for the April 2011 release. Earlier versions are available at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard`. The information in this manual has been updated for new features and functionality introduced in the Serviceguard A.11.20 April 2011 patches, or their superseding patches (PHSS_41628 or later). If the Serviceguard A.11.20 April 2011 or one of its superseding patches has not been installed, some of the information in this manual will not apply, therefore users should refer to the 18th Edition of the Managing Serviceguard manual.

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

- "Serviceguard at a Glance" (page 20), describes a Serviceguard cluster and provides a roadmap for using this guide.
- "Understanding Serviceguard Hardware Configurations" (page 26) provides a general view of the hardware configurations used by Serviceguard.
- "Understanding Serviceguard Software Components" (page 38) describes the software components of Serviceguard and shows how they function within the HP-UX operating system.
- "Planning and Documenting an HA Cluster " (page 88) steps through the planning process and provides a set of worksheets for organizing information about the cluster.
- "Building an HA Cluster Configuration" (page 149) describes the creation of the cluster configuration.
- "Configuring Packages and Their Services " (page 216) describes the creation of high availability packages and the control scripts associated with them.
- "Cluster and Package Maintenance" (page 248) presents the basic cluster administration tasks.
- "Troubleshooting Your Cluster" (page 308) explains cluster testing and troubleshooting strategies.
- "Enterprise Cluster Master Toolkit " (page 326) describes a group of tools to simplify the integration of popular applications with Serviceguard.
- "Designing Highly Available Cluster Applications " (page 327)gives guidelines for creating cluster-aware applications that provide optimal performance in a Serviceguard environment.
- "Integrating HA Applications with Serviceguard" (page 340)," presents suggestions for integrating your existing applications with Serviceguard.
- "Software Upgrades " (page 343) shows how to move from one Serviceguard or HP-UX release to another without bringing down your applications.
- "Blank Planning Worksheets" (page 355)," contains a set of empty worksheets for preparing a Serviceguard configuration.
- "Migrating from LVM to VxVM Data Storage " (page 360)," describes how to convert from LVM data storage to VxVM data storage.
- "IPv6 Network Support" (page 364) describes the IPv6 addressing scheme and the primary/standby interface configurations supported.
- Appendix I (page 371) describes the Serviceguard Manager GUI.
- "Maximum and Minimum Values for Parameters" (page 375) provides a reference to the supported ranges for Serviceguard parameters.

Related Publications

For information about the current version of Serviceguard, and about older versions, see the Serviceguard documents posted at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard`.

The following documents, which can all be found at http://www.hp.com/go/hpux-serviceguard-docs, are particularly useful.

- The latest edition of the *HP Serviceguard Version A.11.20 Release Notes*
- *HP Serviceguard Quorum Server Version A.04.00 Release Notes*
- *Serviceguard Extension for RAC Version A.11.20 Release Notes*
- *Using Serviceguard Extension for RAC*
- *Understanding and Designing Serviceguard Disaster Tolerant Architectures*
- *Designing Disaster Tolerant HA Clusters Using Metrocluster and Continentalclusters*
- *Enterprise Cluster Master Toolkit Version Release Notes*
- *Serviceguard/SGeRAC/SMS/Serviceguard Mgr Plug-in Compatibility and Feature Matrix*.
- *Securing Serviceguard* and other Serviceguard white papers.

For HP-UX system administration information, see the documents at http://www.hp.com/go/hpux-core-docs.

For information on the Distributed Systems Administration Utilities (DSAU), see the latest version of the *Distributed Systems Administration Utilities Release Notes* and the *Distributed Systems Administration Utilities User's Guide* at http://www.hp.com/go/hpux-core-docs: go to the `HP-UX 11i v3` collection and scroll down to `Getting started`.

For information about the Event Monitoring Service, see the following documents at http://www.hp.com/go/hpux-ha-monitoring-docs:

- *Using the Event Monitoring Service*
- *Using High Availability Monitors*

1 Serviceguard at a Glance

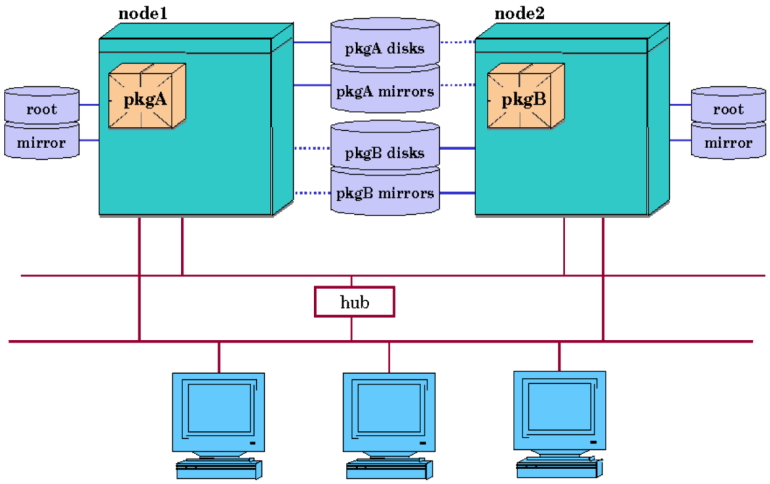
This chapter introduces Serviceguard on HP-UX, and shows where to find information in this book. It covers the following:

- What is Serviceguard?
- Using Serviceguard Manager (page 23)
- A Roadmap for Configuring Clusters and Packages (page 24)

If you are ready to start setting up Serviceguard clusters, skip ahead to Chapter 4: "Planning and Documenting an HA Cluster " (page 88). Specific steps for setup are given in Chapter 5: "Building an HA Cluster Configuration" (page 149).

Figure 1 shows a typical Serviceguard cluster with two nodes.

Figure 1 Typical Cluster Configuration



What is Serviceguard?

Serviceguard allows you to create high availability clusters of HP 9000 or HP Integrity servers (or a mixture of both; see the release notes for your version for details and restrictions).

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 (or both), known as nodes, having sufficient redundancy of software and hardware that a single point of failure will not significantly disrupt service.

A package groups application services (individual HP-UX processes) together. There are failover packages, system multi-node packages, and multi-node packages:

- The typical high availability package is a failover package. It usually is configured to run on several nodes in the cluster, and runs on one at a time. If a service, node, network, or other package resource fails on the node where it is running, Serviceguard can automatically transfer

control of the package to another cluster node, allowing services to remain available with minimal interruption.

- There are also packages that run on several cluster nodes at once, and do not fail over. These are called system multi-node packages and multi-node packages. Examples are the packages HP supplies for use with the Veritas Cluster Volume Manager and Veritas Cluster File System from Symantec (on HP-UX releases that support them; see "About Veritas CFS and CVM from Symantec" (page 22)).

  A system multi-node package must run on all nodes that are active in the cluster. If it fails on one active node, that node halts. System multi-node packages are supported only for HP-supplied applications.

  A multi-node package can be configured to run on one or more cluster nodes. It is considered UP as long as it is running on any of its configured nodes.

In Figure 1, node 1 (one of two SPU's) is running failover 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. In this example, 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.")

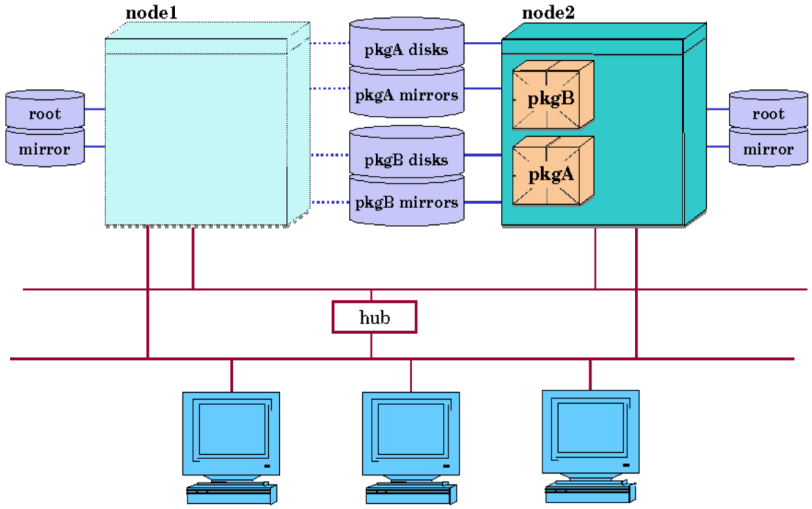
Failover

Any host system running in a Serviceguard cluster is called an active node. Under normal conditions, a fully operating Serviceguard cluster monitors the health of the cluster's components on all its active nodes.

Most Serviceguard packages are failover packages. When you configure a failover package, you specify which active node will be the primary node where the package will start, and one or more other nodes, called adoptive nodes, that can also run the package.

Figure 2 shows what happens in a failover situation.

Figure 2 Typical Cluster After Failover



After this transfer, the failover 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 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;
- HP-supported uninterruptible power supplies (UPS), such as HP PowerTrust, which eliminates failures related to power outage.

HP recommends these products; in conjunction with Serviceguard they provide the highest degree of availability.

About Veritas CFS and CVM from Symantec

Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for Veritas Cluster File System (CFS) and Cluster Volume Manager (CVM): http://www.hp.com/go/hpux-serviceguard-docs.

Using Serviceguard Manager

NOTE: For more-detailed information, see Appendix I (page 371) and the section on Serviceguard Manager in the latest version of the Serviceguard Release Notes. Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for up-to-date information about Serviceguard Manager compatibility. You can find both documents at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard`.

Serviceguard Manager is the graphical user interface for Serviceguard. It is available as a "plug-in" to the System Management Homepage (SMH). SMH is a web-based graphical user interface (GUI) that replaces SAM as the system administration GUI as of HP-UX 11i v3 (but you can still run the SAM terminal interface; see "Using SAM" (page 23)).

You can use Serviceguard Manager to monitor, administer, and configure Serviceguard clusters.

- You can see properties, status, and alerts of clusters, nodes, and packages.
- You can do administrative tasks such as run or halt clusters, cluster nodes, and packages.
- You can create or modify a cluster and its packages.

Monitoring Clusters with Serviceguard Manager

From the main page of Serviceguard Manager, you can see status and alerts for the cluster, nodes, and packages. You can also drill down to see the configuration and alerts of the cluster, nodes, and packages.

Administering Clusters with Serviceguard Manager

Serviceguard Manager allows you administer clusters, nodes, and packages if access control policies permit:

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

Configuring Clusters with Serviceguard Manager

You can configure clusters and packages in Serviceguard Manager. You must have root (UID=0) access to the cluster nodes.

Starting Serviceguard Manager

To start the Serviceguard Manager plug-in in your web browser from the System Management Homepage, click on the link to `Serviceguard Cluster` or a particular cluster. Then select a cluster, node, or package, and use the drop-down menus below the "Serviceguard Manager" banner to navigate to the task you need to do.

Use Serviceguard Manager's built-in help to guide you through the tasks; this manual will tell you if a task can be done in Serviceguard Manager but does not duplicate the help.

Using SAM

You can use SAM, the System Administration Manager, to do many of the HP-UX system administration tasks described in this manual (that is, tasks, such as configuring disks and file systems, that are not specifically Serviceguard tasks).

To launch SAM, enter

```
/usr/sbin/sam
```

on the command line. As of HP-UX 11i v3, SAM offers a Terminal User Interface (TUI) which also acts as a gateway to the web-based System Management Homepage (SMH).

- To get to the SMH for any task area, highlight the task area in the SAM TUI and press **w.**
- To go directly to the SMH from the command line, enter

  /usr/sbin/sam -w

For more information, see the HP-UX *Systems Administrator's Guide*, at the address given in the preface to this manual.

What are the Distributed Systems Administration Utilities?

HP Distributed Systems Administration Utilities (DSAU) simplify the task of managing multiple systems, including Serviceguard clusters. The tools provide:

- Configuration synchronization
- Log consolidation
- Command fan-out

Configuration synchronization allows you to manage configuration changes from a single server, known as the configuration master; your other systems are defined as clients. Changes you make on the configuration master are propagated to all the clients.

Log consolidation allows you to simplify monitoring by consolidating logs from all systems into a single file sorted by time-stamp.

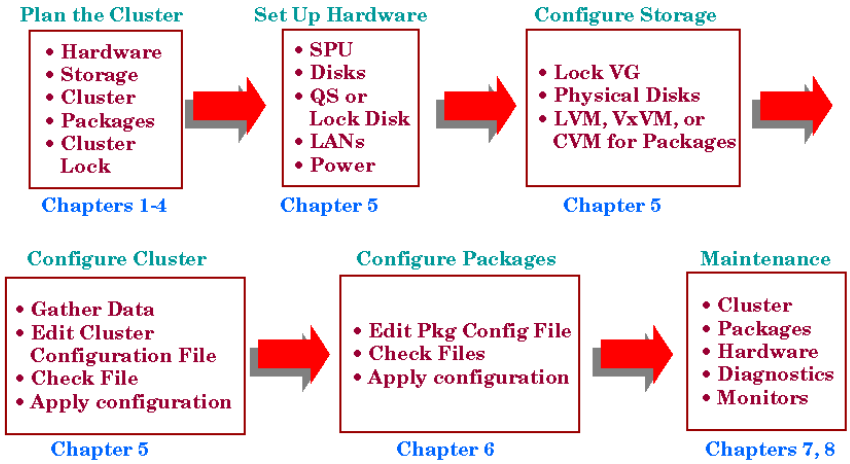
Command fan-out allows you to send the same command to all systems in your configuration in a single action.

For more information on DSAU, see the *Distributed Systems Administration Utilities User's Guide*, at the address given in the preface to this manual.

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 3.

Figure 3 Tasks in Configuring a Serviceguard Cluster



The tasks in Figure 3 are covered in step-by-step detail in chapters 4 through 7. HP recommends you gather all the data that is needed for configuration *before you start*. See "Planning and Documenting an HA Cluster " (page 88) 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 (page 27)
- Redundant Disk Storage (page 31)
- Redundant Power Supplies (page 35)
- Larger Clusters (page 35)

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 a Serviceguard cluster is 16. 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 StorageWorks 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 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 StorageWorks 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. 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. In addition, if a LAN card fails on startup, or cannot be detected because of other hardware failures, Serviceguard will effect a switch to a standby if there is one available in the same bridged net.

Serviceguard supports a maximum of 30 network interfaces per node. For this purpose an interface is defined as anything represented as a LAN interface in the output of `lanscan (1m)`, so the total of 30 can comprise any combination of physical LAN ports, VLAN ports, IPoIB interfaces and APA aggregates. (A node can have more than 30 such interfaces, but only 30 can be part of the cluster configuration.)

A selection of network configurations is described further in the following sections. See also "How the Network Manager Works " (page 64). For detailed information about supported network configurations, consult Hewlett-Packard support.

NOTE: Serial (RS232) lines are no longer supported for the cluster heartbeat.

Fibre Channel, Token Ring and FDDI networks are no longer supported as heartbeat or data LANs.

Rules and Restrictions

- A single subnet cannot be configured on different network interfaces (NICs) on the same node.
- In the case of subnets that can be used for communication between cluster nodes, the same network interface must not be used to route more than one subnet configured on the same node.
- For IPv4 subnets, Serviceguard does not support different subnets on the same LAN interface.
  - For IPv6, Serviceguard supports up to two subnets per LAN interface (site-local and global).
- Serviceguard *does* support different subnets on the same bridged network (this applies at both the node and the cluster level).
- Serviceguard does not support using networking tools such as `ifconfig` or the configuration file `/etc/rc.config.d/netconf` to add IP addresses to network interfaces that are

configured into the Serviceguard cluster, unless those IP addresses themselves will be immediately configured into the cluster as stationary IP addresses.

△ **CAUTION:** If you configure any address other than a stationary IP address on a Serviceguard network interface, it could collide with a relocatable package IP address assigned by Serviceguard. See "Stationary and Relocatable IP Addresses " (page 64).

(Oracle VIPs are an exception to this rule; such configurations require the HP add-on product Serviceguard Extension for Oracle RAC).

- Similarly, Serviceguard does not support using networking tools to move or reconfigure any IP addresses configured into the cluster.

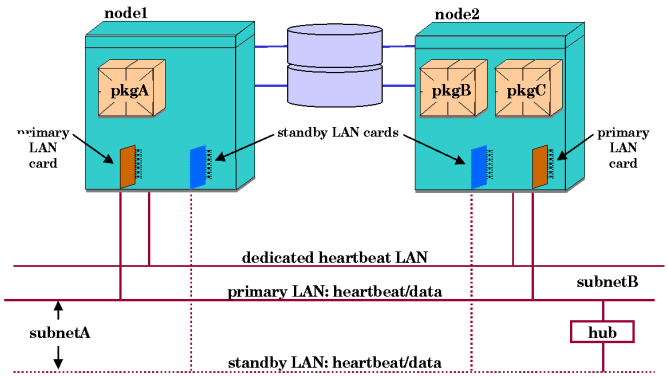
  Doing so leads to unpredictable results because the Serviceguard view of the configuration is different from the reality.

NOTE: If you will be using a cross-subnet configuration, see also the Restrictions (page 29) that apply specifically to such configurations.

Redundant Ethernet Configuration

The use of redundant network components is shown in Figure 4, which is an Ethernet configuration.

Figure 4 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 (`subnetA`). 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 4, 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 the dedicated heartbeat LAN.

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

Cross-Subnet Configurations

As of Serviceguard A.11.18 it is possible to configure multiple subnets, joined by a router, both for the cluster heartbeat and for data, with some nodes using one subnet and some another.

A cross-subnet configuration allows:

- Automatic package failover from a node on one subnet to a node on another
- A cluster heartbeat that spans subnets.

Configuration Tasks

Cluster and package configuration tasks are affected as follows:

- You must use the `-w full` option to `cmquerycl` to discover actual or potential nodes and subnets across routers.
- You must configure two new parameters in the package configuration file to allow packages to fail over across subnets:
  - *`ip_subnet_node`* - to indicate which nodes the subnet is configured on
  - *`monitored_subnet_access`* - to indicate whether the subnet is configured on all nodes (`FULL`) or only some (`PARTIAL`)

  (For legacy packages, see "Configuring Cross-Subnet Failover" (page 295).)
- You should not use the wildcard (`*`) for *`node_name`* in the package configuration file, as this could allow the package to fail over across subnets when a node on the same subnet is eligible, and failing over across subnets can take longer.

  List the nodes in order of preference rather than using the wildcard.
- You should configure IP monitoring for each subnet; see "Monitoring LAN Interfaces and Detecting Failure: IP Level" (page 70).

Restrictions

The following restrictions apply:

- All nodes in the cluster must belong to the same network domain (that is, the domain portion of the fully-qualified domain name must be the same).
- The nodes must be fully connected at the IP level.
- A minimum of two heartbeat paths must be configured for each cluster node.
- There must be less than 200 milliseconds of latency in the heartbeat network.
- Each heartbeat subnet on each node must be physically routed separately to the heartbeat subnet on another node; that is, each heartbeat path must be physically separate:
  - The heartbeats must be statically routed; static route entries must be configured on each node to route the heartbeats through different paths.
  - Failure of a single router must not affect both heartbeats at the same time.
- IPv6 heartbeat subnets are not supported in a cross-subnet configuration.
- IPv6–only and mixed modes are not supported in a cross-subnet configuration. For more information about these modes, see "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102).

- Because Veritas Cluster File System from Symantec (CFS) requires link-level traffic communication (LLT) among the nodes, Serviceguard cannot be configured in cross-subnet configurations with CFS alone.

  But CFS *is* supported in specific cross-subnet configurations with Serviceguard and HP add-on products; see the documentation listed below.

- Each package subnet must be configured with a standby interface on the local bridged net. The standby interface can be shared between subnets.
- You must not set the HP-UX network parameter `ip_strong_es_model` in a cross-subnet configuration. Leave it set to the default (`0`, meaning disabled); Serviceguard does not support enabling it for cross-subnet configurations. For more information about this parameter, see "Tuning Network and Kernel Parameters" (page 162) and "Using a Relocatable Address as the Source Address for an Application that is Bound to INADDR_ANY" (page 335).
- Deploying applications in this environment requires careful consideration; see "Implications for Application Deployment" (page 146).
- `cmrunnode` will fail if the "hostname LAN" is down on the node in question. ("Hostname LAN" refers to the public LAN on which the IP address that the node's hostname resolves to is configured).
- If a *`monitored_subnet`* is configured for `PARTIAL` *`monitored_subnet_access`* in a package's configuration file, it must be configured on at least one of the nodes on the *`node_name`* list for that package. Conversely, if all of the subnets that are being monitored for this package are configured for `PARTIAL` access, each node on the *`node_name`* list must have at least one of these subnets configured.
  - As in other configurations, a package will not start on a node unless the subnets configured on that node, and specified in the package configuration file as monitored subnets, are up.

NOTE: See also the Rules and Restrictions (page 27) that apply to all cluster networking configurations.

For More Information

For more information on the details of configuring the cluster and packages in a cross-subnet context, see "Obtaining Cross-Subnet Information" (page 181), "About Cross-Subnet Failover" (page 145) and (for legacy packages only) "Configuring Cross-Subnet Failover" (page 295).

See also the white paper *Technical Considerations for Creating a Serviceguard Cluster that Spans Multiple IP Subnets*, which you can find at the address below. This paper discusses and illustrates supported configurations, and also potential mis-configurations.

IMPORTANT: Although cross-subnet topology can be implemented on a single site, it is most commonly used by extended-distance clusters, and specifically site-aware disaster-tolerant clusters, which require Metrocluster (HP add-on software).

Design and configuration of such clusters are covered in the disaster-tolerant documentation delivered with Serviceguard. For more information, see the following documents under http://www.hp.com/go/hpux-serviceguard-docs:

- *Understanding and Designing Serviceguard Disaster Tolerant Architectures*
- *Designing Disaster Tolerant HA Clusters Using Metrocluster and Continentalclusters*
- The white paper *Configuration and Administration of Oracle 10g R2 RAC Database in HP Metrocluster*

Replacing 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." With some restrictions, you can also add and delete LAN interfaces to and from the cluster configuration while the cluster is running; see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

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), or Veritas Volume Manager (VxVM) (or Veritas Cluster Volume Manager (CVM). LVM and VxVM disk storage groups can be activated by no more than one node at a time, but when a failover package is moved, the storage group can be activated by the adoptive node. All of the disks in the storage group owned by a failover 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
- 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.

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

Serviceguard itself does not provide protection for data on your disks, but protection is provided by HP's Mirrordisk/UX product for LVM storage, and by the Veritas Volume Manager for VxVM and CVM.

The logical volumes used for Serviceguard packages should be mirrored; so should the cluster nodes' root disks.

When you configure logical volumes using software mirroring, the members of each mirrored set contain exactly the same data. If one disk fails, the storage manager automatically keeps the data available by using the mirror. you can use three-way mirroring in LVM (or additional plexes with VxVM) to allow for online backups or 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. But 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 allows the system to continue normal operation when a root disk failure occurs.

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 protects against single points of failure in the I/O channel, and RAID 1 or 5 configuration provides redundancy for the storage media.

About Multipathing

Multipathing is automatically configured in HP-UX 11i v3 (this is often called native multipathing), or in some cases can be configured with third-party software such as EMC Powerpath.

NOTE: 4.1 and later versions of Veritas Volume Manager (VxVM) and Dynamic Multipathing (DMP) from Symantec are supported on HP-UX 11i v3, but *do not* provide multipathing and load balancing; DMP acts as a pass-through driver, allowing multipathing and load balancing to be controlled by the HP-UX I/O subsystem instead.

For more information about multipathing in HP-UX 11i v3, see the white paper *HP-UX 11i v3 Native Multi-Pathing for Mass Storage*, and the *Logical Volume Management* volume of the HP-UX *System Administrator's Guide* at the address given in the preface to this manual. See also "About Device File Names (Device Special Files)" (page 77).

Monitoring LVM 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 . 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, see "Using EMS to Monitor Volume Groups" (page 96).

Monitoring VxVM and CVM Disks

The HP Serviceguard VxVM Volume Monitor provides a means for effective and persistent monitoring of VxVM and CVM volumes. The Volume Monitor supports Veritas Volume Manager versions 4.1 and 5.0, as well as Veritas Cluster Volume Manager (CVM) versions 4.1and 5.0.

You can configure the Volume Monitor (`cmvxserviced`) to run as a service in a package that requires the monitored volume or volumes. When a monitored volume fails or becomes inaccessible, the service will exit, causing the package to fail on the current node. (The package's failover behavior depends on its configured settings, as with any other failover package.)

For example, the following *`service_cmd`* monitors two volumes at the default log level 0, with a default polling interval of 60 seconds, and prints all log messages to the console:

```
/usr/sbin/cmvxserviced /dev/vx/dsk/cvm_dg0/lvol1 /dev/vx/dsk/cvm_dg0/lvol2
```

For more information, see the `cmvxserviced (1m)` manpage. For more information about configuring package services, see the parameter descriptions starting with *`service_name`* (page 232).

Replacing 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" (page 310) .

Replacing 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" (page 313).

Sample SCSI Disk Configurations

Figure 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 5 Mirrored Disks Connected for High Availability

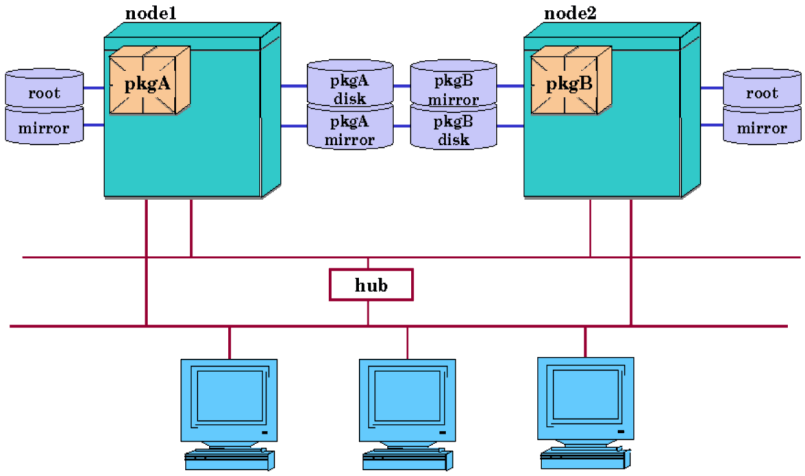
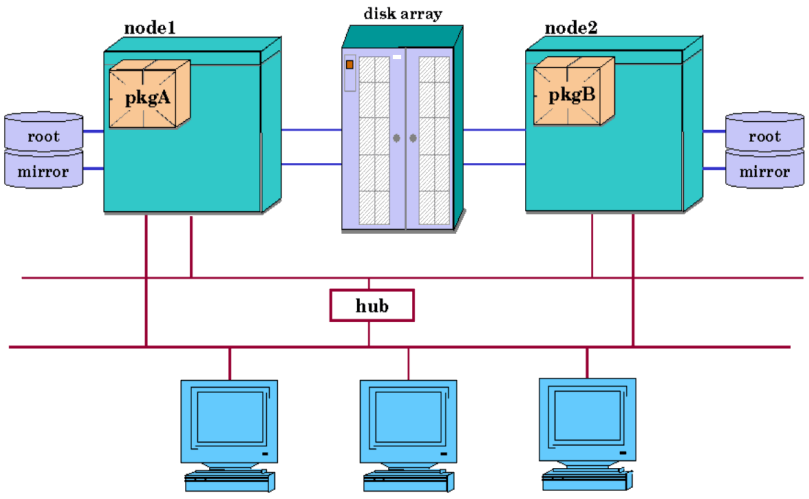


Figure 6 below shows a similar cluster with a disk array connected to each node on two I/O channels. See "About Multipathing" (page 32).

Figure 6 Cluster with High Availability Disk Array



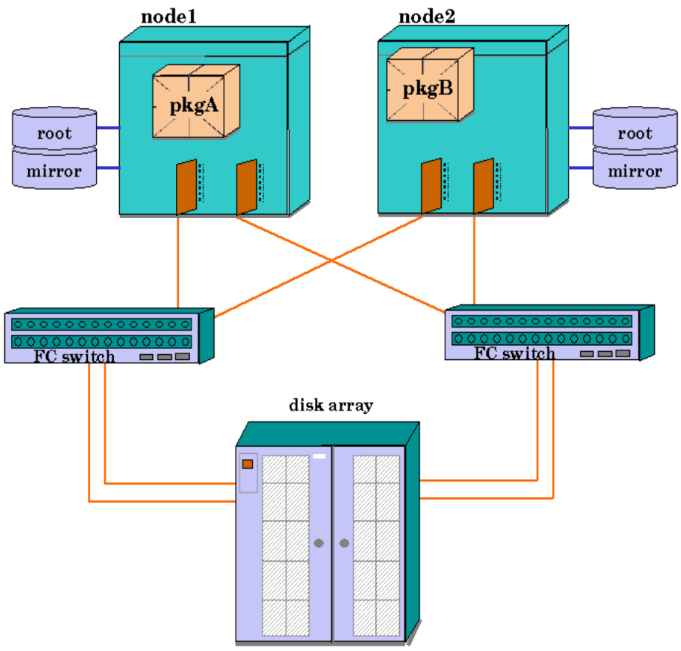
Details on logical volume configuration for Serviceguard are in the chapter "Building an HA Cluster Configuration."

Sample Fibre Channel Disk Configuration

In Figure 7, 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 7 Cluster with Fibre Channel Switched Disk Array



This type of configuration uses native HP-UX or other multipathing software; see "About Multipathing" (page 32).

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 supply other than that used by 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.

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 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.

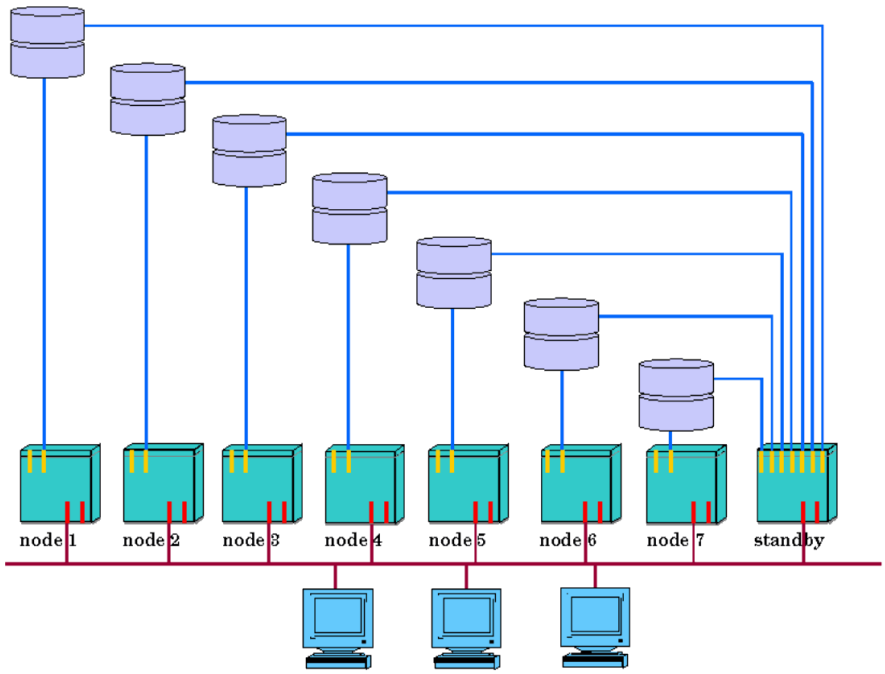
In the case of non-shared SCSI connections to an XP series or EMC disk array, the four-node limit does not apply. Each node can be connected directly to the XP or EMC by means of two SCSI buses. Packages can be configured to fail over among all sixteen nodes. For more about this type of configuration, see "Point to Point Connections to Storage Devices " (page 36).

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. In the man pages for some commands, you can find options to help to reduce the time. For example, refer to the man page for `cmquerycl (1m)` for options that can reduce the amount of time needed for probing disks or networks.

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 8.

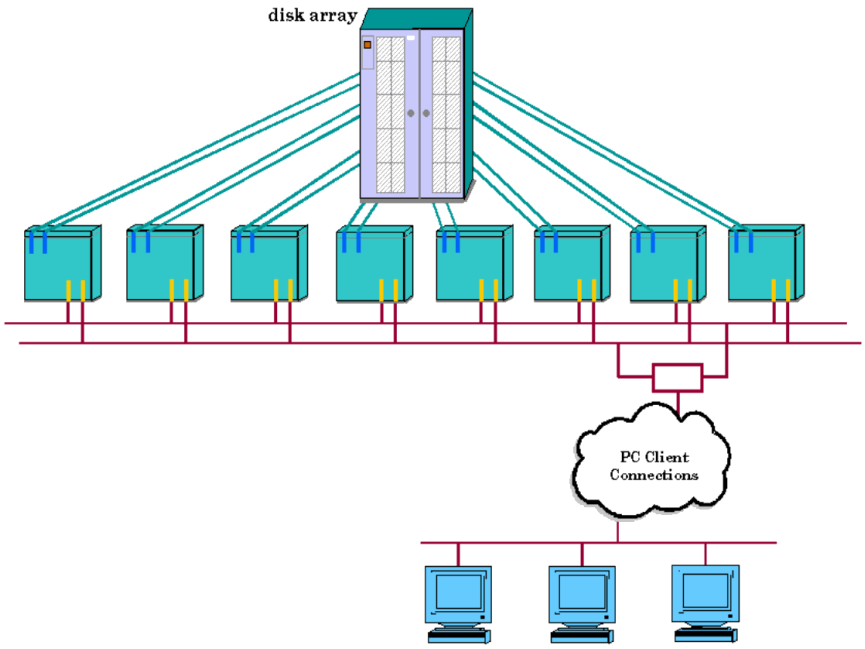
Figure 8 Eight-Node Active/Standby Cluster



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 SCSI interconnect. The nodes access shared data on an XP or EMC disk array configured with 16 SCSI I/O ports. Each node is connected to the array using two separate SCSI channels. Each channel is a dedicated bus; there is no daisy-chaining of the SCSI bus.

Figure 9 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 (page 42)
- How the Package Manager Works (page 48)
- How Packages Run
- How the Network Manager Works (page 64)
- Volume Managers for Data Storage
- Responses to Failures (page 85)

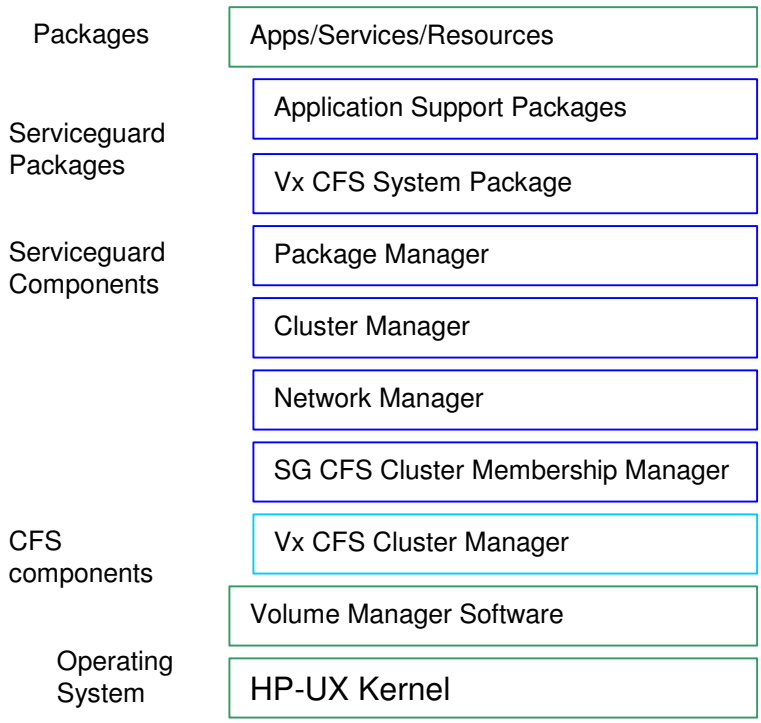
If you are ready to start setting up Serviceguard clusters, skip ahead to Chapter 5: "Building an HA Cluster Configuration" (page 149).

Serviceguard Architecture

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

NOTE: Veritas CFS may not yet be supported on the version of HP-UX you are running; see "About Veritas CFS and CVM from Symantec" (page 22).

Figure 10 Serviceguard Software Components



Serviceguard Daemons

Serviceguard uses the following daemons:

- `/usr/lbin/cmclconfd`—Serviceguard Configuration Daemon
- `/usr/lbin/cmcld`—Serviceguard Cluster Daemon

- /usr/lbin/cmfileassistd—Serviceguard File Management 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/cmserviced—Serviceguard Service Assistant Daemon
- /usr/lbin/qs—Serviceguard Quorum Server Daemon
- /usr/lbin/cmnetd—Serviceguard Network Manager daemon.
- /usr/lbin/cmvxd—Serviceguard-to-Veritas Membership Coordination daemon. (Only present if Veritas CFS is installed.)
- /usr/lbin/cmvxpingd—Serviceguard-to-Veritas Activation daemon. (Only present if Veritas CFS is installed.)
- /usr/lbin/cmdisklockd— Lock LUN daemon
- /usr/lbin/cmlockd—Utility daemon
- /opt/sgproviders/bin/cmwbemd—WBEM daemon
- /usr/lbin/cmproxyd—Proxy daemon

Each of these daemons logs to the /var/adm/syslog/syslog.logfile except for /opt/cmom/lbin/cmomd, which logs to /var/opt/cmom/cmomd.log. The Quorum Server runs outside the cluster. By default, it logs to the standard output, and it is suggested you redirect output to a file named/var/adm/qs/qs.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 determines cluster membership by sending heartbeat messages to cmcld daemons on other nodes in 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 (Transfer of Control) or INIT, which is an immediate system reset without a graceful shutdown. (This manual normally refers to this event simply as a system reset.) 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 system reset resulting from the expiration of the safety timer, messages will be written to /var/adm/syslog/syslog.log and the kernel's message buffer, and a system dump is performed.

The duration of the safety timer depends on the cluster configuration parameter *MEMBER_TIMEOUT*, and also on the characteristics of the cluster configuration, such as whether it uses a Quorum Server or a cluster lock (and what type of lock) and whether or not standby LANs are configured.

For further discussion, see "What Happens when a Node Times Out" (page 85). For advice on setting *MEMBER_TIMEOUT*, see "Cluster Configuration Parameters " (page 105). For troubleshooting, see "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320).

cmcld also manages Serviceguard packages, determining where to run them and when to start them.

NOTE: Two of the central components of Serviceguard—Package Manager, and Cluster 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 priority lower than 20, otherwise they may prevent Serviceguard from updating the kernel safety timer, causing a system reset.

File Management Daemon: cmfileassistd

The `cmfileassistd` daemon is used by `cmcld` to manage the files that it needs to read from, and write to, disk. This is to prevent any delays in issuing Input/Output from impacting the timing of `cmcld`.

Syslog Log Daemon: cmlogd

`cmlogd` is used by `cmcld` to write messages to `syslog`. Any message written to `syslog` by `cmcld` is written via `cmlogd`. This is to prevent any delays in writing to `syslog` from affecting 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 that depend on knowledge of the state of cluster objects.

Clients send queries to the object manager and receive responses from it (this communication is done indirectly, through a Serviceguard API). 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 a particular query.

This daemon is started by `inetd(1M)`. There are entries in the `/etc/inetd.conf` file.

This daemon may not be running on your system; it is used only by clients of the object manager.

Cluster SNMP Agent Daemon: cmsnmpd

This daemon collaborates with the SNMP Master Agent to provide instrumentation for the cluster Management Information Base (MIB).

The SNMP Master Agent and the `cmsnmpd` provide notification (traps) for cluster-related events. For example, a trap is sent when the cluster configuration changes, or when a Serviceguard package has failed. You must edit `/etc/SnmpAgent.d/snmpd.conf` to tell `cmsnmpd` where to send this information.

You must also edit `/etc/rc.config.d/cmsnmpagt` to auto-start `cmsnmpd`. Configure `cmsnmpd` to start before the Serviceguard cluster comes up.

For more information, see the `cmsnmpd (1m)` manpage.

Service Assistant Daemon: cmserviced

This daemon forks and execs any scripts or processes 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 cmserviced or it causes the package to halt and moves the package to an available alternate node.

Quorum Server Daemon: qs

Using a Quorum Server is one way to break a tie and establish a quorum when the cluster is re-forming; the other way is to use a cluster lock. See "Cluster Quorum to Prevent Split-Brain Syndrome" and "Cluster Lock " (page 44).

The Quorum Server, if used, runs on a system external to the cluster and is started by the system administrator, not by Serviceguard. It is normally started from /etc/inittab with the respawn option, which means that it automatically restarts if it fails or is killed. All members of the cluster initiate and maintain a connection to the Quorum Server; if it dies, the Serviceguard nodes will detect this and then periodically try to reconnect to it. If there is a cluster re-formation while the Quorum Server is down and tie-breaking is needed, the re-formation will fail and all the nodes will halt (system reset). For this reason it is important to bring the Quorum Server back up as soon as possible.

For more information about the Quorum Server software and how it works, see the latest version of the *HP Serviceguard Quorum Server* release notes at http://www.hp.com/go/hpux-serviceguard-docs under HP Serviceguard Quorum Server Software

Network Manager Daemon: cmnetd

This daemon monitors the health of cluster networks, and performs local LAN failover. It also handles the addition and deletion of relocatable package IP addresses for both IPv4 and IPv6.

Lock LUN Daemon: cmdisklockd

If a lock LUN is being used, cmdisklockd runs on each node in the cluster and is started by cmcld when the node joins the cluster.

Utility Daemon: cmlockd

Runs on every node on which cmcld is running (though currently not actually used by Serviceguard on HP-UX systems).

Cluster WBEM Agent Daemon: cmwbemd

This daemon collaborates with the Serviceguard WBEM provider plug-in module (SGProviders) and WBEM services cimserver to provide notification (WBEM Indications) of Serviceguard cluster events to Serviceguard WBEM Indication subscribers that have registered a subscription with the cimserver. For example, an Indication is sent when the cluster configuration changes, or when a Serviceguard package has failed.

You must edit /etc/rc.config.d/cmwbemagt to auto-start cmwbemd. Configure cmwbemd to start before the Serviceguard cluster comes up. You can start and stop cmwbemd with the commands /sbin/init.d/cmwbemagt start and /sbin/init.d/cmwbemagt stop.

Proxy Daemon: cmproxyd

This daemon is used to proxy or cache Serviceguard configuration data for use by certain Serviceguard commands running on the local node. This allows these commands to get the data quicker and removes the burden of responding to certain requests from cmcld.

CFS Components

The HP Serviceguard Storage Management Suite offers additional components for interfacing with the Veritas Cluster File System on some current versions of HP-UX (see "About Veritas CFS and CVM from Symantec" (page 22)). Documents for the management suite are posted on http://www.hp.com/go/hpux-serviceguard-docs.

Veritas CFS components operate directly over Ethernet networks that connect the nodes within a cluster. Redundant networks are required to avoid single points of failure.

The Veritas CFS components are:

- GAB (Group Membership Services/Atomic Broadcast) — When Veritas Cluster Volume Manager (CVM) 4.1 or later, or Veritas Cluster File System (CFS), is deployed as part of the Serviceguard Storage Management Suite bundles, the file `/etc/gabtab` is automatically configured and maintained by Serviceguard.

  GAB provides membership and messaging for CVM and the CFS. GAB membership also provides orderly startup and shutdown of the cluster file system.

- LLT (Low Latency Transport) - When Veritas CVM or CFS is deployed as part of the Serviceguard Storage Management Suite bundles, the LLT files`/etc/llthosts` and `/etc/lltab` are automatically configured and maintained by Serviceguard.

  LLT provides kernel-to-kernel communications and monitors network communications for CFS.

- vxfend - When Veritas CFS is deployed as part of the Serviceguard Storage Management Suite, the I/O fencing daemon `vxfend` is also included. It implements a quorum-type functionality for the Veritas Cluster File System. `vxfend` is controlled by Serviceguard to synchronize quorum mechanisms.
- cmvxd - The Serviceguard-to-Veritas daemon coordinates the membership information between Serviceguard and Veritas' Clustered File System product. (Only present when Veritas CFS is installed.)
- cmvxping- The Serviceguard-to-Veritas daemon activates certain subsystems of the Veritas Clustered File System product. (Only present when Veritas CFS is installed.)

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.

Configuring 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 the chapter "Planning and Documenting an HA Cluster " (page 88)). You can set cluster parameters using Serviceguard Manager or by editing the cluster configuration file (see Chapter 5: "Building an HA Cluster Configuration" (page 149)). 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 UDP heartbeat messages with every other node over each monitored IP network configured as a heartbeat device. (LAN monitoring is discussed later, in the section "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66).)

If a cluster node does not receive heartbeat messages from all other cluster nodes within the prescribed time, a cluster re-formation is initiated; see "What Happens when a Node Times Out"

(page 85) . At the end of the re-formation, information about the new cluster membership is passed to the package coordinator (described further in this chapter, in "How the Package Manager Works" (page 48)). Failover packages that were running on nodes that are no longer in the new cluster are transferred to their adoptive nodes.

If heartbeat and data are sent over the same LAN subnet, data congestion may cause Serviceguard to miss heartbeats and initiate a cluster re-formation that would not otherwise have been needed. For this reason, HP recommends that you dedicate a LAN for the heartbeat as well as configuring heartbeat over the data network.

NOTE: You can no longer run the heartbeat on a serial (RS232) line or an FDDI or Token Ring network.

Each node sends its heartbeat message at a rate calculated by Serviceguard on the basis of the value of the *MEMBER_TIMEOUT* parameter, set in the cluster configuration file, which you create as a part of cluster configuration.

IMPORTANT: When multiple heartbeats are configured, heartbeats are sent in parallel; Serviceguard must receive at least one heartbeat to establish the health of a node. HP recommends that you configure all subnets that connect cluster nodes as heartbeat networks; this increases protection against multiple faults at no additional cost.

Heartbeat IP addresses are usually on the same subnet on each node, but it is possible to configure a cluster that spans subnets; see "Cross-Subnet Configurations" (page 29).

For more information about heartbeat requirements, see the entry for *HEARTBEAT_IP*, under "Cluster Configuration Parameters ". For timeout requirements and recommendations, see the *MEMBER_TIMEOUT* parameter description in the same section. For troubleshooting information, see "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320). See also "Cluster Daemon: cmcld" (page 39).

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 `/etc/rc.config.d/cmcluster`. 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 by means of a lock disk, lock LUN, 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 (system reset). 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 three nodes, a cluster lock is strongly recommended. If you have a cluster with more than four nodes, use a Quorum Server; a cluster lock disk is not allowed for clusters of that size.

Use of a Lock LUN or LVM Lock Disk as the Cluster Lock

A lock disk or lock LUN can be used for clusters up to and including four nodes in size.

A 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. Similarly, a cluster lock LUN is a small dedicated LUN, connected to all nodes in the cluster, that contains the lock information.

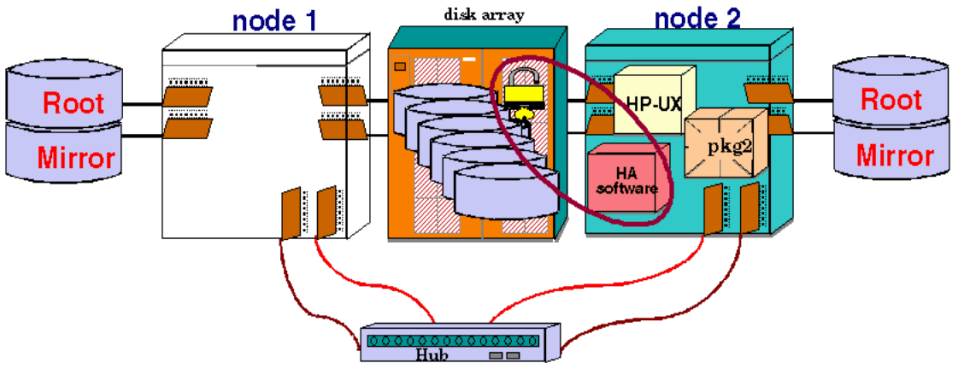
In an LVM configuration, a disk used as a lock disk is not dedicated for use as the cluster lock; the disk can be employed as part of a normal volume group with user data on it. A lock LUN, on the other hand, is dedicated to the cluster lock; you cannot store any other data on it.

You specify the cluster lock volume group and physical volume, or the cluster lock LUN, in the cluster configuration file.

When a node obtains the cluster lock, this area is marked so that other nodes will recognize the lock as "taken."

The operation of the lock disk or lock LUN is shown in Figure 11.

Figure 11 Lock Disk or Lock LUN Operation



Serviceguard periodically checks the health of the lock disk or LUN and writes messages to the `syslog` file if the device fails the health check. This file should be monitored for early detection of lock disk problems.

If you are using a lock disk, 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.

IMPORTANT: A dual lock cannot be implemented on LUNs. This means that the lock LUN mechanism cannot be used in an Extended Distance cluster.

Single Lock Disk or LUN

A single lock disk or lock LUN should be configured on a power circuit separate from that of any node in the cluster. For example, using three power circuits for a two-node cluster is highly recommended, with a separately powered disk or LUN for the cluster lock. In two-node clusters, this single lock device must not share a power circuit with either node, and a lock disk 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, 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.

NOTE: You must use Fibre Channel connections for a dual cluster lock; you can no longer implement it in a parallel SCSI configuration.

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 if the remaining node(s) cannot access the surviving cluster lock disk.

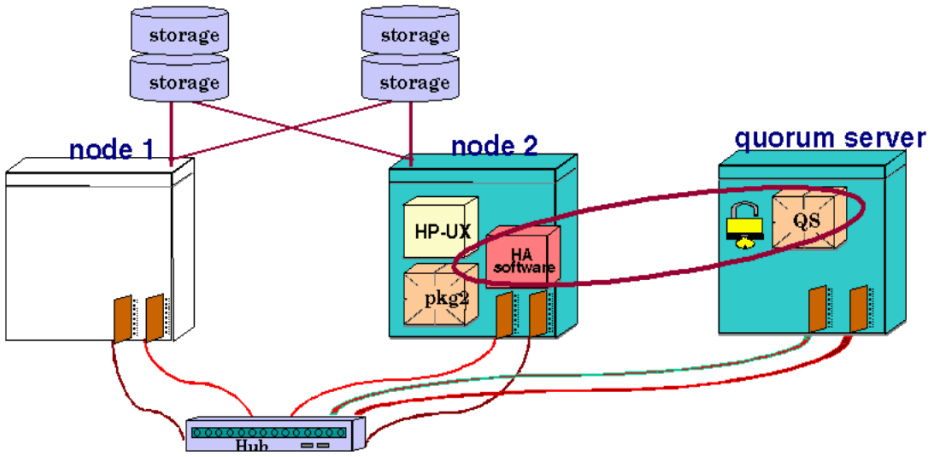
Use of the Quorum Server as the Cluster Lock

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 system reset. 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 when its arbitration services are needed, the cluster will halt.

The operation of the Quorum Server is shown in Figure 12. 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 12 Quorum Server Operation



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

IMPORTANT: For more information about the Quorum Server, see the latest version of the *HP Serviceguard Quorum Server* release notes at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard Quorum Server Software`

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 re-forms 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, and that you don't have exactly half of the nodes on a single power circuit.

What Happens when You Change the Quorum Configuration Online

You can change the quorum configuration while the cluster is up and running. This includes changes to the quorum method (for example from a lock disk to a Quorum Server), the quorum device (for example from one quorum server to another), and the parameters that govern them (for example the Quorum Server polling interval). For more information about the Quorum Server and lock parameters, see "Cluster Configuration Parameters " (page 105).

When you make quorum configuration changes, Serviceguard goes through a two-step process:

1. All nodes switch to a strict majority quorum (turning off any existing quorum devices).
2. All nodes switch to the newly configured quorum method, device and parameters.

IMPORTANT: During Step 1, while the nodes are using a strict majority quorum, node failures can cause the cluster to go down unexpectedly if the cluster has been using a quorum device before the configuration change. For example, suppose you change the Quorum Server polling interval while a two-node cluster is running. If a node fails during Step 1, the cluster will lose quorum and go down, because a strict majority of prior cluster members (two out of two in this case) is required. The duration of Step 1 is typically around a second, so the chance of a node failure occurring during that time is very small.

In order to keep the time interval as short as possible, make sure you are changing only the quorum configuration, and nothing else, when you apply the change.

If this slight risk of a node failure leading to cluster failure is unacceptable, halt the cluster before you make the quorum configuration change.

How the Package Manager Works

Packages are the means by which Serviceguard starts and halts configured applications. A package is a collection of services, disk volumes and IP addresses that are managed by Serviceguard to ensure they are available.

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 control scripts that run and halt packages and their services.
- Reacts to changes in the status of monitored resources.

Package Types

Three different types of packages can run in the cluster; the most common is the failover package. There are also special-purpose packages that run on more than one node at a time, and so do not failover. They are typically used to manage resources of certain failover packages.

Non-failover Packages

There are two types of special-purpose packages that do not fail over and that can run on more than one node at the same time: the system multi-node package, which runs on all nodes in the cluster, and the multi-node package, which can be configured to run on all or some of the nodes in the cluster. System multi-node packages are reserved for use by HP-supplied applications, such as Veritas Cluster Volume Manager (CVM) and Cluster File System (CFS).

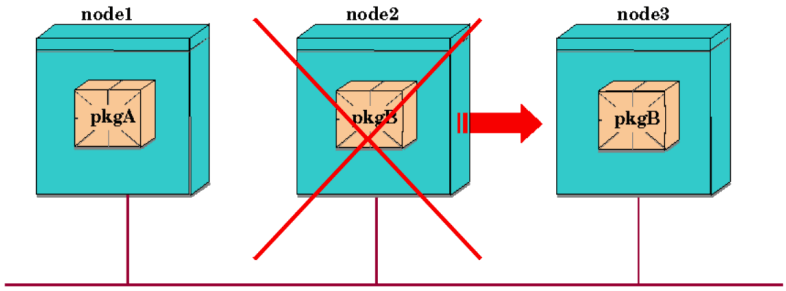
The rest of this section describes failover packages.

Failover Packages

A failover package starts up on an appropriate node (see `node_name` on (page 223)) 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 13 Package Moving During Failover



Configuring Failover Packages

You configure each package separately. You create a failover package by generating and editing a package configuration file template, then adding the package to the cluster configuration database; see Chapter 6: "Configuring Packages and Their Services " (page 216).

For legacy packages (packages created by the method used on versions of Serviceguard earlier than A.11.18), you must also create a package control script for each package, to manage the execution of the package's services. See "Configuring a Legacy Package" (page 289) for detailed information.

Customized package control scripts are not needed for modular packages (packages created by the method introduced in Serviceguard A.11.18). These packages are managed by a master control script that is installed with Serviceguard; see Chapter 6: "Configuring Packages and Their Services " (page 216), for instructions for creating modular packages.

Deciding When and Where to Run and Halt Failover Packages

The package configuration file assigns a name to the package and includes a list of the nodes on which the package can run.

Failover packages list the nodes in order of priority (i.e., the first node in the list is the highest priority node). In addition, failover packages' files contain three parameters that determine failover behavior. These are the *auto_run* parameter, the *failover_policy* parameter, and the *failback_policy* parameter.

Failover Packages' Switching Behavior

The *auto_run* parameter (known in earlier versions of Serviceguard as the *PKG_SWITCHING_ENABLED* parameter) defines the default global switching attribute for a failover package at cluster startup: that is, whether Serviceguard can automatically start the package when the cluster is started, and whether Serviceguard should automatically restart the package on a new node in response to a failure. Once the cluster is running, the package switching attribute of each package can be temporarily set with the `cmmodpkg` command; at reboot, the configured value will be restored.

The *auto_run* parameter is set in the package configuration file.

A package switch normally involves moving a failover package and its associated IP addresses to a new system on the same subnet. In this case, the new system must have the same subnet configured and working properly; otherwise the package will not be started.

NOTE: It is possible to configure a cluster that spans subnets joined by a router, with some nodes using one subnet and some another. This is known as a cross-subnet configuration. In this context, you can configure packages to fail over from a node on one subnet to a node on another, and you will need to configure a relocatable IP address for each subnet the package is configured to start on; see "About Cross-Subnet Failover" (page 145), and in particular the subsection "Implications for Application Deployment" (page 146).

When a package fails over, 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, normally all of them must be available on the target node before the package will be started. (In a cross-subnet configuration, all the monitored subnets that are specified for this package, and configured on the target node, must be up.)

If the package has a dependency on a resource or another package, the dependency must be met on the target node before the package can start.

The switching of relocatable IP addresses on a single subnet is shown in Figure 14 and Figure 15. Figure 14 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 the 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 14 Before Package Switching

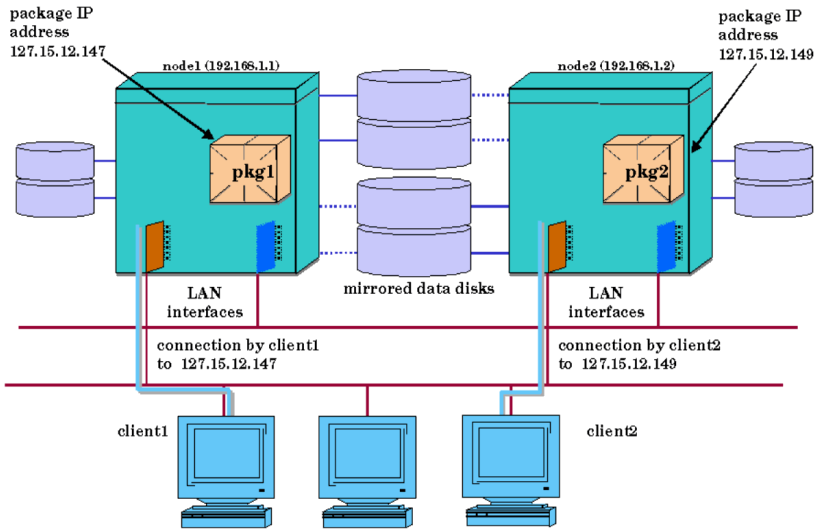
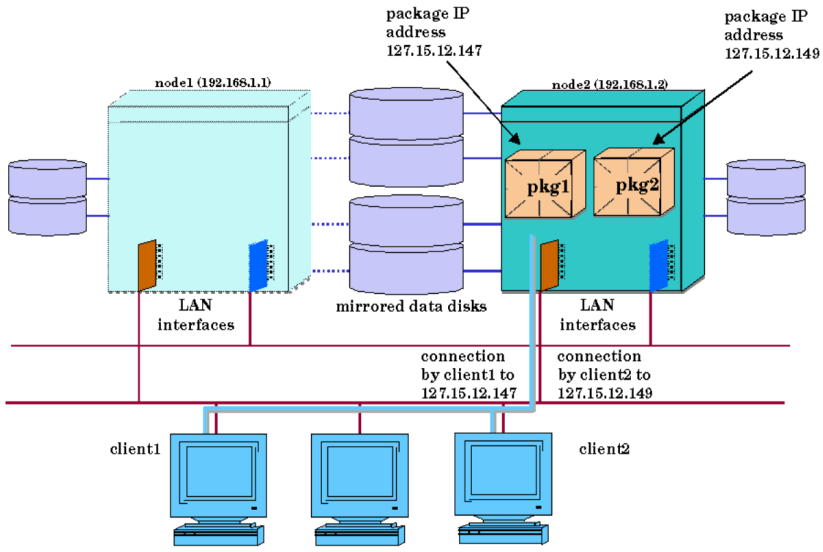


Figure 15 shows the condition where Node 1 has failed and Package 1 has been transferred to Node 2 on the same subnet. Package 1's IP address was transferred to Node 2 along with the package. Package 1 continues to be available and is now running on Node 2. Also note that Node 2 can now access both Package1's disk and Package2's disk.

NOTE: For design and configuration information about clusters which span subnets, including site-aware disaster-tolerant clusters, see the documents listed under "Cross-Subnet Configurations" (page 29).

Figure 15 After Package Switching



Failover Policy

The Package Manager selects a node for a failover package to run on based on the priority list included in the package configuration file together with the *failover_policy* parameter, also in the configuration file. The 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 failover policies are `configured_node` (the default), `min_package_node`, `site_preferred` and `site_preferred_manual`. The parameter is set in the package configuration file. See "*failover_policy*" (page 226) for more information.

Package placement is also affected by package dependencies and weights, if you choose to use them. See "About Package Dependencies" (page 128) and "About Package Weights" (page 135).

Automatic Rotating Standby

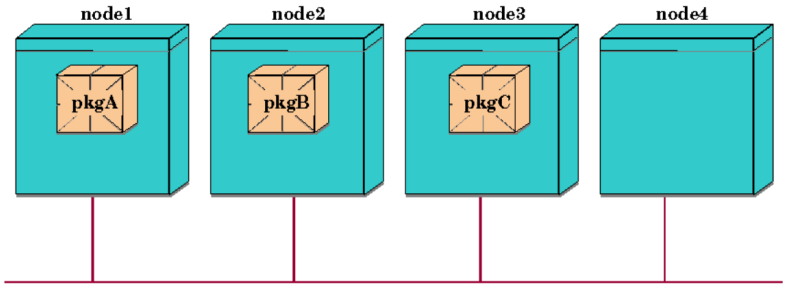
Using the `min_package_node` failover policy, it is possible to configure a cluster that lets you use one node as an automatic rotating standby node for the cluster. Consider the following package configuration for a four node cluster. Note that all packages can run on all nodes and have the same `node_name`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

Table 2 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`

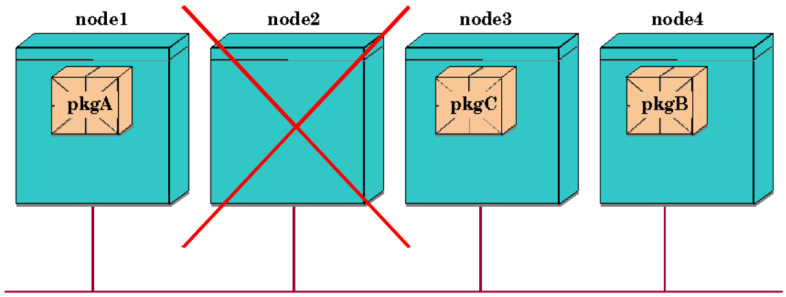
When the cluster starts, each package starts as shown in Figure 16.

Figure 16 Rotating Standby Configuration before Failover



If a failure occurs, any package would fail over to the node containing fewest running packages, as in Figure 17, which shows a failure on node 2:

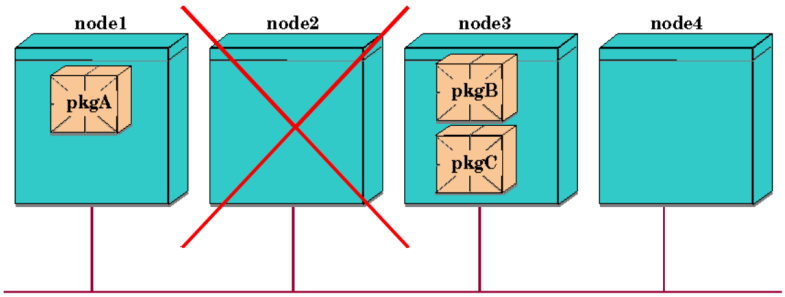
Figure 17 Rotating Standby Configuration after Failover



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 16, but the failure of node 2 would cause the package to start on node 3, as in Figure 18:

Figure 18 `CONFIGURED_NODE` Policy Packages after Failover



If you use `configured_node` as 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 set in the package configuration file:

As an example, consider the following four-node configuration, in which *`failover_policy`* is set to `configured_node` and *`failback_policy`* is `automatic`:

Figure 19 Automatic Failback Configuration before Failover

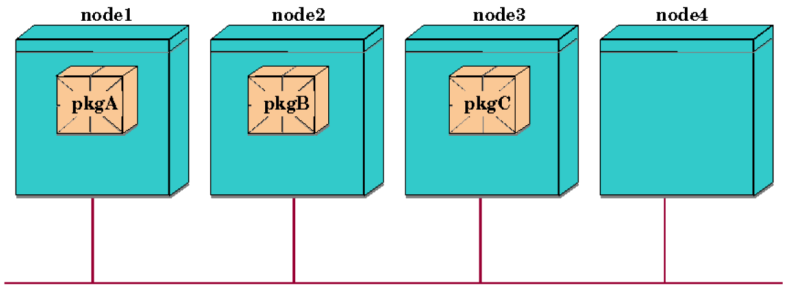
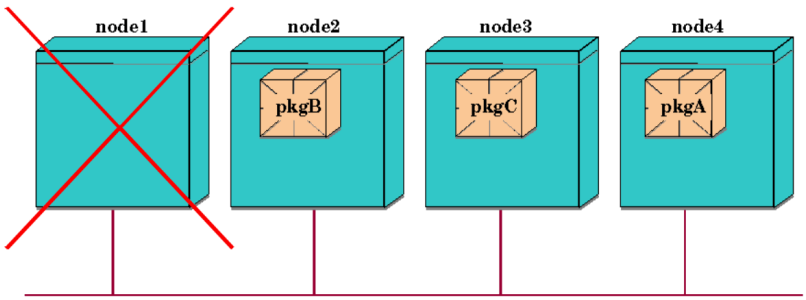


Table 3 Node Lists in Sample Cluster

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

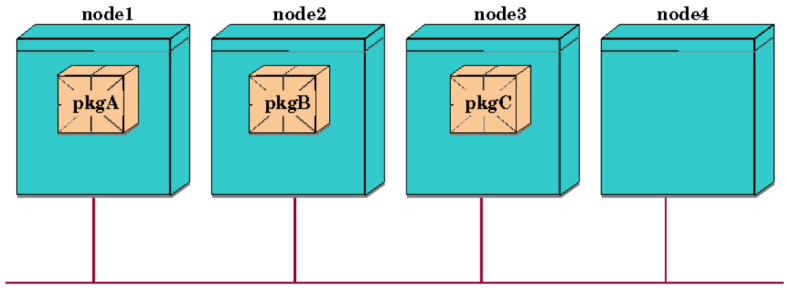
node1 panics, and after the cluster reforms, pkgA starts running on node4:

Figure 20 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 21 Automatic Failback Configuration After Restart of Node 1



△ **CAUTION:** 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 want to wait to add the package's primary node back into the cluster until you can allow the package to be taken out of service temporarily while it switches back to the primary node.

Serviceguard automatically chooses a primary node for a package when the *NODE_NAME* is set to '*'. When you set the *NODE_NAME* to '*' and the *failback_policy* is automatic, if you add, delete, or rename a node in the cluster, the primary node for the package might change resulting in the automatic failover of that package.

Using Older Package Configuration Files

If you are using package configuration files that were generated using a previous version of Serviceguard, HP recommends you use the cmmakepkg command to open a new template, and then copy the parameter values into it. In the new template, read the descriptions and defaults of the choices that did not exist when the original configuration was made. For example, the default for *failover_policy* is now configured_node and the default for *failback_policy* is now manual.

For full details of the current parameters and their default values, see Chapter 6: "Configuring Packages and Their Services " (page 216), and the package configuration file template itself.

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 (Event Monitoring Service) High Availability Monitors, and an ATM monitor.

If a monitored 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 monitored resource for a package in Serviceguard Manager, or on the HP-UX command line by using the command `/opt/resmon/bin/resls`. For additional information, see the manpage for `resls(1m)`.

Using the EMS HA Monitors

The EMS (Event Monitoring Service) HA Monitors, available as a separate product, can be used to set up monitoring of disks and other resources as package resource 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 resource 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 for graphical display or for operator notification. For more information, see the latest Event Monitoring Service release notes and other documents at http://www.hp.com/go/hpux-ha-monitoring-docs.

See also "Using EMS to Monitor Volume Groups" (page 96).

How Packages Run

Packages are the means by which Serviceguard starts and halts configured applications. Failover 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 300 packages per cluster and a total of 900 services per cluster.

What Makes a Package Run?

There are 3 types of packages:

- The failover package is the most common type of package. It runs on one node at a time. If a failure occurs, it can switch to another node listed in its configuration file. If switching is enabled for several nodes, the package manager will use the failover policy to determine where to start the package.
- A system multi-node package runs on all the active cluster nodes at the same time. It can be started or halted on all nodes, but not on individual nodes.
- A multi-node package can run on several nodes at the same time. If *auto_run* is set to `yes`, Serviceguard starts the multi-node package on all the nodes listed in its configuration file. It can be started or halted on all nodes, or on individual nodes, either by user command (`cmhaltpkg`) or automatically by Serviceguard in response to a failure of a package component, such as service, EMS resource, or subnet.

System multi-node packages are supported only for use by applications supplied by Hewlett-Packard.

A failover package can be configured to have a dependency on a multi-node or system multi-node package. The package manager cannot start a package on a node unless the package it depends on is already up and running on that node.

The package manager will always try to keep a failover package running unless there is something preventing it from running on any node. The most common reasons for a failover package not being able to run are that *auto_run* is disabled so Serviceguard is not allowed to start the package, that node switching is disabled for the package on particular nodes, or that the package has a dependency that is not being met. Package weights and node capacities, if you choose to use them, also determine whether a package can run on a given node. When a package has failed on one node and is enabled to switch to another node, it will start up automatically in a new location where its dependencies and all other necessary conditions are met. This process is known as package switching, or remote switching.

A failover package starts on the first available node in its configuration file; by default, it fails over to the next available one in the list. Note that you do not necessarily have to use a `cmrunpkg` command to restart a failed failover package; in many cases, the best way is to enable package and/or node switching with the `cmmodpkg` command.

When you create the package, you indicate the list of nodes on which it is allowed to run. System multi-node packages must list all cluster nodes in their cluster. Multi-node packages and failover packages can name some subset of the cluster's nodes or all of them.

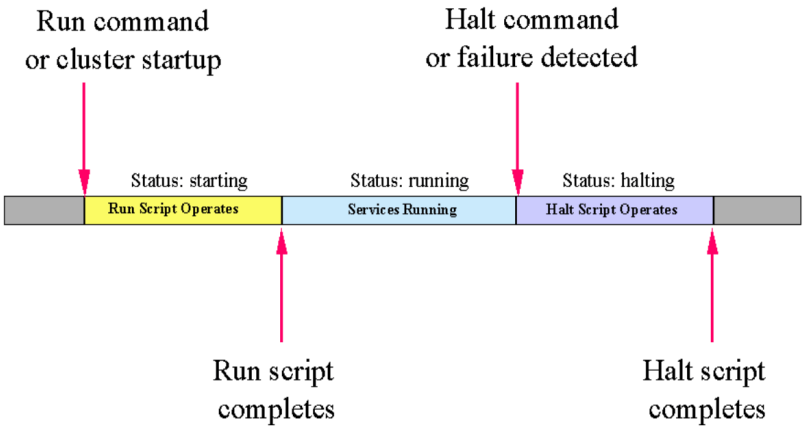
If the *auto_run* parameter is set to `yes` in a package's configuration file Serviceguard automatically starts the package when the cluster starts. System multi-node packages are required to have *auto_run* set to `yes`. If a failover package has *auto_run* set to `no`, Serviceguard cannot start it automatically at cluster startup time; you must explicitly enable this kind of package using the `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, issue the `cmrunpkg` or `cmmodpkg` command.

How does a failover package start up, and what is its behavior while it is running? Some of the many phases of package life are shown in Figure 22.

NOTE: This diagram applies specifically to legacy packages. Differences for modular scripts are called out below.

Figure 22 Legacy Package Time Line Showing Important Events



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

1. Before the control script starts. (For modular packages, this is the master control script.)
2. During run script execution. (For modular packages, during control script execution to start the package.)
3. While services are running
4. When a service, subnet, or monitored resource fails, or a dependency is not met.
5. During halt script execution. (For modular packages, during control script execution to halt the package.)
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 or on a special-purpose package. If monitoring shows a value for a configured resource that is outside the permitted range, the package cannot start.

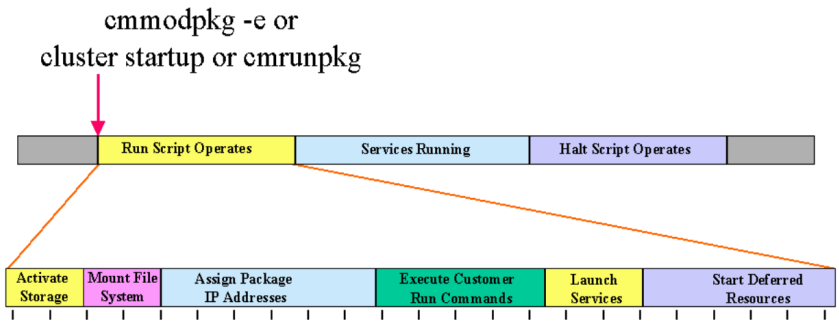
Once a node is selected, a check is then done to make sure the node allows the package to start on it. Then services are started up for a package by the control script on the selected node. Strictly speaking, the run script on the selected node is used to start a legacy package; the master control script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

During Run Script Execution

Once the package manager has determined that the package can start on a particular node, it launches the script that starts the package (that is, a package's control script or master control script is executed with the `start` parameter). This script carries out the following steps:

1. Executes any `external_pre_scripts` (modular packages only (page 239))
2. Activates volume groups or disk groups.
3. Mounts file systems.
4. Assigns package IP addresses to the LAN card on the node (failover packages only).
5. Executes any customer-defined run commands (legacy packages only; see "Adding Customer Defined Functions to the Package Control Script " (page 293)) or `external_scripts` (modular packages only; (page 239)).
6. Starts each package service.
7. Starts up any EMS (Event Monitoring Service) resources needed by the package that were specially marked for deferred startup.
8. Exits with an exit code of zero (0).

Figure 23 Package Time Line (Legacy Package)



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.

NOTE: This diagram is specific to legacy packages. Modular packages also run external scripts and "pre-scripts" as explained above.

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. For legacy packages, this is in the same directory as the run script and has the same name as the run script and the extension `.log`. For modular packages, the pathname is determined by the `script_log_file parameter` in the package configuration file (see (page 225)). 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_fail_fast_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. The package script may not have been able to clean up some of its resources such as LVM volume groups, VxVM disk groups or package mount points, so before attempting to start up the package on any node, be sure to check whether any resources for the package need to be cleaned up.

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. You can configure a number of restarts for each service. 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 in a legacy package; for more information about configuring services in modular packages, see the discussion starting with *service_name* (page 232), and the comments in the package configuration template file.

```
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_fail_fast_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_fail_fast_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, or a Dependency is Not Met

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, if a configured resource fails, or if a configured dependency on a special-purpose package is not met, then a failover package will halt on its current node and, depending on the setting of the package switching flags, may be restarted on another node. If a multi-node or system multi-node package fails, all of the packages that have configured a dependency on it will also fail.

Package halting normally means that the package halt script executes (see the next section). However, if a failover package's configuration has the *service_fail_fast_enabled* flag 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 fails on the node where the package is running, the package will start to shut down. If the subnet recovers immediately (before the package is restarted on an adoptive node), the package manager restarts the package on the same node; no package switch occurs.

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 (see *auto_run* on (page 224)).

You cannot halt a multi-node or system multi-node package unless all packages that have a configured dependency on it are down. Use cmviewcl to check the status of dependents. For example, if pkg1 and pkg2 depend on PKGa, both pkg1 and pkg2 must be halted before you can halt PKGa.

NOTE: If you use cmhaltpkg command with the *-n <nodename>* option, the package is halted only if it is running on that node.

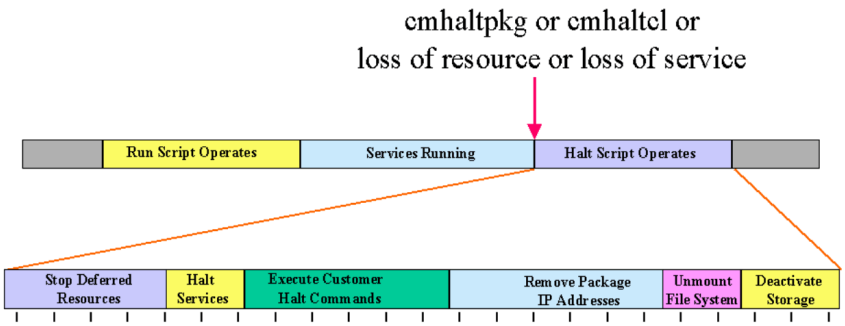
Thecmmodpkg command cannot be used to halt a package, but it can disable switching either on particular nodes or on all nodes. A package can continue running when its switching has been disabled, but it will not be able to start on other nodes if it stops running on its current node.

During Halt Script Execution

Once the package manager has detected the failure of a service or package that a failover package depends on, or when the cmhaltpkg command has been issued for a particular package, the package manager launches the halt script. That is, a package's control script or master control script is executed with the *stop* parameter. This script carries out the following steps (also shown in Figure 24):

1. Halts any deferred resources that had been started earlier.
2. Halts all package services.
3. Executes any customer-defined halt commands (legacy packages only) or *external_script*s (modular packages only(page 239)).
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).
8. Executes any *external_pre_scripts* (modular packages only (page 239)).

Figure 24 Legacy 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. For legacy packages, this is in the same directory as the run script and has the same name as the run script and the extension .log. For modular packages, the pathname is determined by the *script_log_file parameter* in the package configuration file(page 225). Normal starts are recorded in the log, together with error messages or warnings related to halting the package.

NOTE: This diagram applies specifically to legacy packages. Differences for modular scripts are called out above.

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. The package script may not have been able to clean up some of its resources such as LVM volume groups, VxVM disk groups or package mount points, so before attempting to start up the package on any node, be sure to check whether any resources for the package need to be cleaned up

Package Control Script Error and Exit Conditions

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

Table 4 Error Conditions and Package Movement for Failover Packages

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	Either Setting	YES	system reset	No	N/A (system reset)	Yes
Service Failure	Either Setting	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	system reset	No	N/A (system reset)	Yes
Run Script Exit 2	NO	Either Setting	Running	No	No	Yes
Run Script Timeout	YES	Either Setting	system reset	No	N/A (system reset)	Yes
Run Script Timeout	NO	Either Setting	Running	No	Not changed	No
Halt Script Exit 1	YES	Either Setting	Running	N/A	Yes	No
Halt Script Exit 1	NO	Either Setting	Running	N/A	Yes	No
Halt Script Timeout	YES	Either Setting	system reset	N/A	N/A (system reset)	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	system reset	No	N/A (system reset)	Yes
Service Failure	Either Setting	NO	Running	Yes	No	Yes

Table 4 Error Conditions and Package Movement for Failover Packages *(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 Network	YES	Either Setting	system reset	No	N/A (system reset)	Yes
Loss of Network	NO	Either Setting	Running	Yes	Yes	Yes
Loss of Monitored Resource	YES	Either Setting	system reset	No	N/A (system reset)	Yes
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
dependency package failed	Either Setting	Either Setting	Running	Yes	Yes when dependency is again met	Yes if dependency met

How the Network Manager Works

The purpose of the network manager is to detect and recover from network card 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.

NOTE: Serviceguard monitors the health of the network interfaces (NICs) and can monitor the IP level (layer 3) network.

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.

IMPORTANT: Every subnet configured as a *`monitored_subnet`* in a package configuration file must be configured into the cluster via *`NETWORK_INTERFACE`* and either *`STATIONARY_IP`* or *`HEARTBEAT_IP`* in the cluster configuration file. See "Cluster Configuration Parameters " (page 105) and "Package Parameter Explanations" (page 222) for more information.

In addition to the stationary IP address, you normally assign one or more unique IP addresses to each failover 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 on the same subnet. You can use up to 200 relocatable IP addresses in a cluster. These addresses can be IPv4, IPv6, or a combination of both address families.

Because system multi-node and multi-node packages do not fail over, they do not have relocatable IP address.

A relocatable IP address is like a virtual host IP address that is assigned to a package. HP recommends that you configure names for each package through DNS (Domain Name Service). A program can then 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 IP addresses, and relocatable IP addresses on subnets that are configured into the cluster, 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 on the same subnet 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.

IMPORTANT: Any subnet that is used by a package for relocatable addresses should be configured into the cluster via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP* in the cluster configuration file. For more information about those parameters, see "Cluster Configuration Parameters " (page 105). For more information about configuring relocatable addresses, see the descriptions of the package *ip_* parameters (page 231).

NOTE: It is possible to configure a cluster that spans subnets joined by a router, with some nodes using one subnet and some another. This is called a cross-subnet configuration. In this context, you can configure packages to fail over from a node on one subnet to a node on another, and you will need to configure a relocatable address for each subnet the package is configured to start on; see "About Cross-Subnet Failover" (page 145), and in particular the subsection "Implications for Application Deployment" (page 146).

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. You can define heartbeat IPs, stationary IPs, and relocatable (package) IPs as IPv4 or IPv6 addresses (or certain combinations of both).

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. These functions are performed by the `cmmodnet` command in the package master control script (package control script for legacy packages).

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.

CAUTION: HP strongly recommends that you add relocatable addresses to packages *only* by editing *ip_address* (page 232) in the package configuration file (or *IP []* entries in the control script of a legacy package) and running `cmapplyconf (1m)`.

Load Sharing

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

Monitoring LAN Interfaces and Detecting Failure: Link Level

At regular intervals, determined by the *NETWORK_POLLING_INTERVAL* (see "Cluster Configuration Parameters " (page 105)) 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 under "Redundant Network Components " (page 27) 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. See "Reporting Link-Level and IP-Level Failures" (page 73).

Serviceguard Network Manager also looks at the numerical counts of packets sent 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 the following values for the *NETWORK_FAILURE_DETECTION* parameter:

NOTE: For a full discussion, see the white paper *Serviceguard Network Manager: Inbound Failure Detection Enhancement* at http://www.hp.com/go/hpux-serviceguard-docs.

- `INOUT`: When both the inbound and outbound counts stop incrementing for a certain amount of time, Serviceguard will declare the card as bad. (Serviceguard calculates the time depending on the type of LAN card.) 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 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.

NOTE: You can change the value of the *NETWORK_FAILURE_DETECTION* parameter while the cluster is up and running.

Local Switching

A local network switch involves the detection of a local network interface failure and a failover to the local backup LAN card (also known as the standby 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 uses 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 is supported in the following configurations:

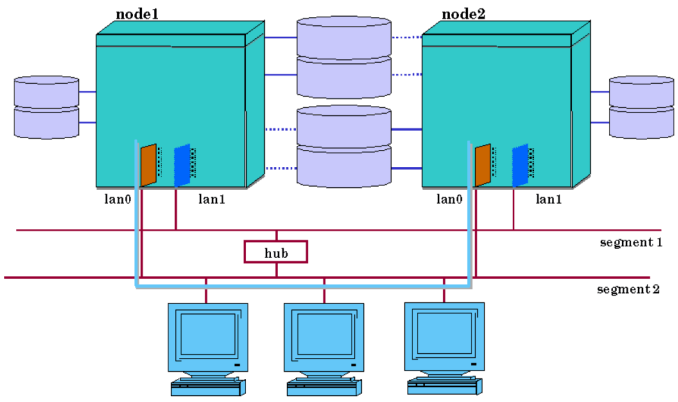
- 1000Base-SX and 1000Base-T
- 1000Base-T or 1000BaseSX and 100Base-T

On HP-UX 11i, however, Jumbo Frames can only be used when the 1000Base-T or 1000Base-SX cards are configured. The 100Base-T and 10Base-T cards do not support Jumbo Frames. Additionally, 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 IPoIB interfaces is not supported, but a local switchover between 10BT Ethernet and 100BT Ethernet is supported.

Figure 25 shows two nodes connected in one bridged net. LAN segments 1 and 2 are connected by a hub.

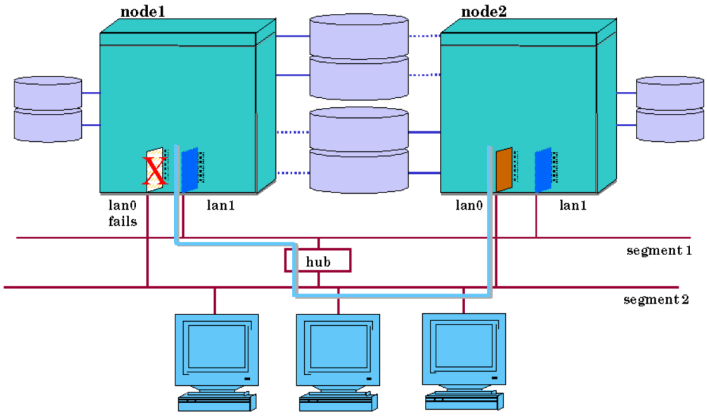
Figure 25 Cluster Before Local Network Switching



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

In Figure 26, we see what would happen if the LAN segment 2 network interface card on Node 1 were to fail.

Figure 26 Cluster After Local Network Switching

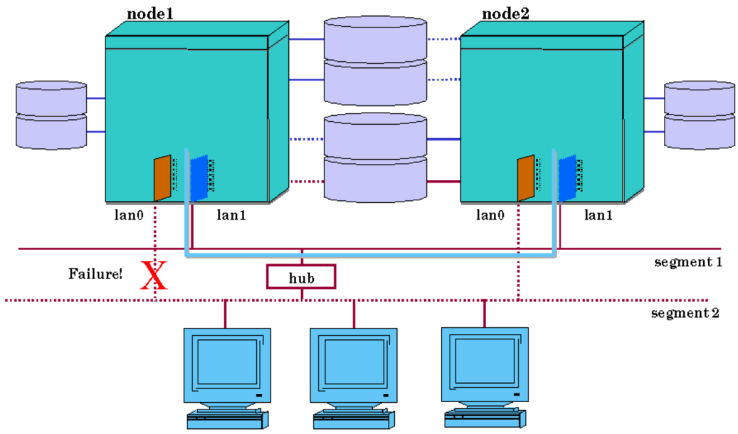


As the standby interface takes over, IP addresses will be switched to the hardware path associated with the standby interface. The switch is transparent at the TCP/IP level. All applications continue to run on their original nodes. During this time, IP traffic on Node 1 will be delayed as the transfer 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 27. In this case a failure affecting segment 2 causes both nodes to switch to the LAN cards attached to segment 1.

Figure 27 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 advantage 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

If a primary interface fails, the IP address will be switched to a standby. You can use the *NETWORK_AUTO_FAILBACK* parameter in the cluster configuration file to configure what Serviceguard does if the primary interface is later restored.

- *Default behavior:* *NETWORK_AUTO_FAILBACK* = YES. Serviceguard will detect and log the recovery of the interface, and will switch the IP address back to its primary interface.

  In addition, when a node is halted, the cluster daemon (cmcld) will attempt to switch any Serviceguard-configured IP addresses on standby interfaces back to their primary interfaces, 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.

- *Configurable behavior:* *NETWORK_AUTO_FAILBACK* = NO. Serviceguard will detect and log the recovery of the interface, but will not switch the IP address back from the standby to the primary interface.

  You can tell Serviceguard to switch the IP address back to the primary interface by means of the cmmodnet command:

  cmmodnet -e *<interface>*

  where *<interface>* is the primary interface.

The purpose of the *NETWORK_AUTO_FAILBACK* parameter is to allow you control Serviceguard's behavior if you find network interfaces are experiencing a “ping-pong” effect, switching back and forth unduly between primary and standby interfaces. If you are not seeing any such problem, leave *NETWORK_AUTO_FAILBACK* set to YES. You can track switching behavior in the syslog file.

NOTE: The *NETWORK_AUTO_FAILBACK* setting applies only to link-level failures, not to failures at the IP level; see “Monitoring LAN Interfaces and Detecting Failure: IP Level” (page 70) for more information about such failures. For more information about the cluster configuration file, see “Cluster Configuration Parameters ” (page 105).

Remote Switching

A remote switch (that is, a package switch) involves moving packages to a new system. In the most common configuration, in which all nodes are on the same subnet(s), the package IP (relocatable IP; see “Stationary and Relocatable IP Addresses ” (page 64)) moves as well, and the new system must already have the subnet configured and working properly, otherwise the packages will not be started.

NOTE: It is possible to configure a cluster that spans subnets joined by a router, with some nodes using one subnet and some another. This is called a cross-subnet configuration. In this context, you can configure packages to fail over from a node on one subnet to a node on another, and you will need to configure a relocatable address for each subnet the package is configured to start on; see “About Cross-Subnet Failover” (page 145), and in particular the subsection “Implications for Application Deployment” (page 146).

When a remote switch occurs, 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 (specified as *monitored_subnet*s in the package configuration file), all those subnets must normally be available on the target node before the package will be started. (In a cross-subnet

configuration, all subnets configured on that node, and identified as monitored subnets in the package configuration file, must be available.)

Note that remote switching is supported only between LANs of the same type. For example, a remote switchover between an Ethernet interface on one machine and an IPoIB interface on the failover machine is not supported. The remote switching of relocatable IP addresses is shown in Figure 14 and Figure 15.

Address Resolution Messages after Switching on the Same Subnet

When a relocatable 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 relocatable 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 their neighbors.

Monitoring LAN Interfaces and Detecting Failure: IP Level

In addition to monitoring network interfaces at the link level, Serviceguard can also monitor the IP level, checking Layer 3 health and connectivity for both IPv4 and IPv6 subnets. This is done by the IP Monitor, which is configurable: you can enable IP monitoring for any subnet configured into the cluster, but you do not have to monitor any. You can configure IP monitoring for a subnet, or turn off monitoring, while the cluster is running.

The IP Monitor:

- Detects when a network interface fails to send or receive IP messages, even though it is still able to send and/or receive DLPI messages.
- Handles the failure, failover, recovery, and failback.

Reasons To Use IP Monitoring

Beyond the capabilities already provided by link-level monitoring, IP monitoring can:

- Monitor network status beyond the first level of switches; see "How the IP Monitor Works" (page 71)
- Detect and handle errors such as:
  - IP packet corruption on the router or switch
  - Link failure between switches and a first-level router
  - Inbound failures even when the cluster configuration parameter *NETWORK_FAILURE_DETECTION* is not set to INONLY_OR_INOUT

    NOTE: This applies only to subnets for which the cluster configuration parameter *IP_MONITOR* is set to ON. See "Cluster Configuration Parameters " (page 105) for more information.

  - Errors that prevent packets from being received but do not affect the link-level health of an interface

IMPORTANT: You should configure the IP Monitor in a cross-subnet configuration, because IP monitoring will detect some errors that link-level monitoring will not. See also "Cross-Subnet Configurations" (page 29).

How the IP Monitor Works

Using Internet Control Message Protocol (ICMP) and ICMPv6, the IP Monitor sends polling messages to target IP addresses and verifies that responses are received. When the IP Monitor detects a failure, it marks the network interface down at the IP level, as shown in the output of `cmviewcl (1m)`. If there is a standby, the subnet is failed over to the standby. See "Reporting Link-Level and IP-Level Failures" (page 73) and "Failure and Recovery Detection Times" (page 72).

The monitor can perform two types of polling:

- Peer polling.

  In this case the IP Monitor sends ICMP `ECHO` messages from each IP address on a subnet to all other IP addresses on the same subnet on other nodes in the cluster.

- Target polling.

  In this case the IP Monitor sends ICMP `ECHO` messages from each IP address on a subnet to an external IP address specified in the cluster configuration file; see *`POLLING_TARGET`* under "Cluster Configuration Parameters " (page 105). `cmquerycl (1m)` will detect gateways available for use as polling targets, as shown in the example below.

  Target polling enables monitoring beyond the first level of switches, allowing you to detect if the route is broken anywhere between each monitored IP address and the target.

  NOTE: In a cross-subnet configuration, nodes can configure peer interfaces on nodes on the other routed subnet as polling targets.

HP recommends that you configure target polling if the subnet is not private to the cluster.

The IP Monitor section of the `cmquerycl` output looks similar to this:

```
…
Route Connectivity (no probing was performed):

IPv4:

1   16.89.143.192
    16.89.120.0

…

Possible IP Monitor Subnets:

IPv4:

16.89.112.0         Polling Target 16.89.112.1

IPv6:

3ffe:1000:0:a801::  Polling Target 3ffe:1000:0:a801::254

…
```

The IP Monitor section of the cluster configuration file will look similar to the following for a subnet on which IP monitoring is configured with target polling.

NOTE: This is the default if `cmquerycl` detects a gateway for the subnet in question; see *SUBNET* under "Cluster Configuration Parameters " (page 105) for more information.

IMPORTANT: By default, `cmquerycl` does not verify that the gateways it detects will work correctly for monitoring. But if you use the `-w full` option, `cmquerycl` will validate them as polling targets.

```
SUBNET 192.168.1.0
 IP_MONITOR ON
 POLLING_TARGET 192.168.1.254
```

To configure a subnet for IP monitoring with peer polling, edit the IP Monitor section of the cluster configuration file to look similar to this:

```
SUBNET 192.168.2.0
 IP_MONITOR ON
```

NOTE: This is not the default. If `cmquerycl` does not detect a gateway for the subnet in question, it sets *IP_MONITOR* to `OFF`, disabling IP-level polling for this subnet; if it *does* detect a gateway, it populates *POLLING_TARGET*, enabling target polling. See *SUBNET* under "Cluster Configuration Parameters " (page 105) for more information.

The IP Monitor section of the cluster configuration file will look similar to the following in the case of a subnet on which IP monitoring is disabled:

```
SUBNET 192.168.3.0
 IP_MONITOR OFF
```

NOTE: This is the default if `cmquerycl` does not detect a gateway for the subnet in question; it is equivalent to having no *SUBNET* entry for the subnet. See *SUBNET* under "Cluster Configuration Parameters " (page 105) for more information.

Failure and Recovery Detection Times

With the default *NETWORK_POLLING_INTERVAL* of 2 seconds (see "Cluster Configuration Parameters " (page 105)) the IP monitor will detect IP failures typically within 8–10 seconds for Ethernet and within 16–18 seconds for InfiniBand. Similarly, with the default *NETWORK_POLLING_INTERVAL*, the IP monitor will detect the recovery of an IP address typically within 8–10 seconds for Ethernet and with 16–18 seconds for InfiniBand.

IMPORTANT: HP strongly recommends that you do not change the default *NETWORK_POLLING_INTERVAL* value of 2 seconds.

See also "Reporting Link-Level and IP-Level Failures" (page 73).

Constraints and Limitations

- A subnet must be configured into the cluster in order to be monitored.
- Polling targets are not detected beyond the first-level router.
- Polling targets must accept and respond to ICMP (or ICMPv6) `ECHO` messages.
- A peer IP on the same subnet should not be a polling target because a node can always ping itself.
- On an HP-UX system, an IP-level failure on the standby interface will not be detected until the IP address fails over to the standby.
- On an HP-UX system, after the IP address fails over to the standby, the IP monitor will not detect recovery on the primary interface.

The following constraints apply to peer polling when there are only two interfaces on a subnet:

- If one interface fails, both interfaces and the entire subnet will be marked down on each node, unless Local Switching (page 66) is configured and there is a working standby.
- If the node that has one of the interfaces goes down, the subnet on the other node will be marked down.

Reporting Link-Level and IP-Level Failures

Serviceguard detects both link-level and IP-level failures; see "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66) and "Monitoring LAN Interfaces and Detecting Failure: IP Level" (page 70) for information about each level of monitoring. Any given failure may occur at the link level or the IP level. In addition, local switching may occur, that is, a switch to a standby LAN card if one has been configured; see "Local Switching " (page 66).

The following examples show when and how a link-level failure is differentiated from an IP-level failure in the output of `cmviewcl (1m)`.

As you can see, if local switching is configured, the difference is the keyword `disabled`, which appears in the tabular output, and is set to `true` in the line output, if the IP Monitor detects the failure. If local switching is not configured, the output is the same whether link-level or IP monitoring detects the failure.

Example 1: If Local Switching is Configured

If local switching is configured and a failure is detected by link-level monitoring, output from `cmviewcl -v` will look like something like this:

```
Network_Parameters:
    INTERFACE    STATUS                      PATH                NAME
    PRIMARY      down (Link and IP)          0/3/1/0             lan2
    PRIMARY      up                          0/5/1/0             lan3
```

`cmviewcl -v -f line` will report the same failure like this:

```
node:gary|interface:lan2|status=down
node:gary|interface:lan2|local_switch_peer=lan1
node:gary|interface:lan2|disabled=false
node:gary|interface:lan2|failure_type=link+ip
```

But if the IP Monitor detects the failure (and link-level monitoring does not) `cmviewcl -v` output will look something like this:

```
Network_Parameters:
    INTERFACE    STATUS                      PATH                NAME
    PRIMARY      down (disabled) (IP only)   0/3/1/0             lan2
    PRIMARY      up                          0/5/1/0             lan3
```

`cmviewcl -v -f line` will report the same failure like this:

```
node:gary|interface:lan2|status=down
node:gary|interface:lan2|local_switch_peer=lan1
node:gary|interface:lan2|disabled=true
node:gary|interface:lan2|failure_type=ip_only
```

In this case, you would need to re-enable the primary interface on each node after the link is repaired, using `cmmodnet (1m)`; for example:

```
cmmodnet –e lan2
```

Example 2: If There Is No Local Switching

If local switching is not configured and a failure is detected by link-level monitoring, output from `cmviewcl -v` will look like something like this:

```
Network_Parameters:
    INTERFACE    STATUS                      PATH                NAME
```

```
    PRIMARY      down (Link and IP)         0/3/1/0             lan2
    PRIMARY      up                         0/5/1/0             lan3
```

`cmviewcl -v -f line` will report the same failure like this:

```
node:gary|interface:lan2|status=down
node:gary|interface:lan2|disabled=false
node:gary|interface:lan2|failure_type=link+ip
```

If local switching is not configured and a failure is detected by IP monitoring, output from `cmviewcl -v` will look like something like this:

```
Network_Parameters:
    INTERFACE    STATUS                     PATH                NAME
    PRIMARY      down (IP only)             0/3/1/0             lan2
    PRIMARY      up                         0/5/1/0             lan3
```

`cmviewcl -v -f line` will report the same failure like this:

```
node:gary|interface:lan2|status=down
node:gary|interface:lan2|disabled=false
node:gary|interface:lan2|failure_type=ip_only
```

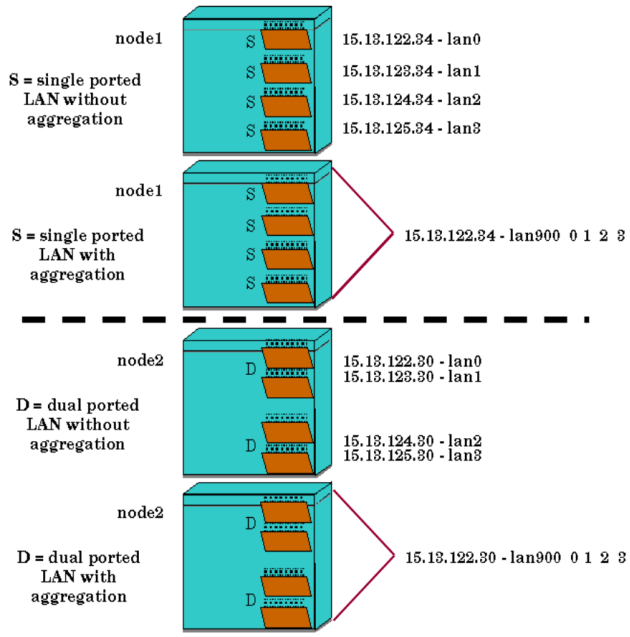
Automatic Port Aggregation

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 32 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. Alternatively, you can aggregate ports from different cards. Figure 28 shows two examples.

Figure 28 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.

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 APAs 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 the latest version of the *HP Auto Port Aggregation (APA) Administrator's Guide* and other APA documents posted at http://www.hp.com/go/hpux-networking-docs.

VLAN Configurations

Virtual LAN configuration using HP-UX VLAN software is supported in Serviceguard clusters.

What is VLAN?

Virtual LAN (or VLAN) is a technology that allows logical grouping of network nodes, regardless of their physical locations.

VLAN can be used to divide a physical LAN into multiple logical LAN segments or broadcast domains, helping to reduce broadcast traffic, increase network performance and security, and improve manageability.

Multiple VLAN interfaces, each with its own IP address, can be configured from a physical LAN interface; these VLAN interfaces appear to applications as ordinary network interfaces (NICs). See *Using HP-UX VLAN* for more information on configuring VLAN interfaces.

Support for HP-UX VLAN

VLAN interfaces can be used as heartbeat as well as data networks in the cluster. The Network Manager monitors the health of VLAN interfaces configured in the cluster, and performs local and

remote failover of VLAN interfaces when failure is detected. Failure of a VLAN interface is typically the result of the failure of the underlying physical NIC port or aggregated (APA) ports.

Configuration Restrictions

HP-UX allows up to 1024 VLANs to be created from a physical NIC port. A large pool of system resources is required to accommodate such a configuration; Serviceguard could suffer performance degradation if many network interfaces are configured in each cluster node. To prevent this and other problems, Serviceguard imposes the following restrictions:

- A maximum of 30 network interfaces per node is supported. The interfaces can be physical NIC ports, VLAN interfaces, APA aggregates, or any combination of these.
- Local failover of VLANs must be onto the same link types. For example, you must fail over from VLAN-over-Ethernet to VLAN-over-Ethernet.
- The primary and standby VLANs must have same VLAN ID (or tag ID).
- VLAN configurations are only supported on HP-UX 11i releases.
- Only port-based and IP-subnet-based VLANs are supported. Protocol-based VLAN is not 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, unless it is a standby for a primary VLAN interface.
- Failover from physical LAN interfaces to VLAN interfaces or vice versa is not supported because of restrictions in VLAN software.
- Using VLAN in a Wide Area Network cluster is not supported.
- If CVM disk groups are used, you must not configure the Serviceguard heartbeat over VLAN interfaces.

Additional Heartbeat Requirements

VLAN technology allows great flexibility in network configuration. To maintain Serviceguard's reliability and availability in such an environment, the heartbeat rules are tightened as follows when the cluster is using VLANs:

1. VLAN heartbeat networks must be configured on separate physical NICs or APA aggregates, to avoid single points of failure.
2. Heartbeats are still recommended on all cluster networks, including VLANs.
3. If you are using VLANs, but decide not to use VLANs for heartbeat networks, heartbeats are recommended for all other physical networks or APA aggregates specified in the cluster configuration 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. Two types of mirroring are RAID1 and RAID5. 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 hardware RAID1 or RAID5.

About Device File Names (Device Special Files)

HP-UX releases up to and including 11i v2 use a naming convention for device files that encodes their hardware path. For example, a device file named `/dev/dsk/c3t15d0` would indicate SCSI controller instance 3, SCSI target 15, and SCSI LUN 0. HP-UX 11i v3 introduces a new nomenclature for device files, known as agile addressing (sometimes also called persistent LUN binding).

Under the agile addressing convention, the hardware path name is no longer encoded in a storage device's name; instead, each device file name reflects a unique instance number, for example `/dev/[r]disk/disk3`, that does not need to change when the hardware path does.

Agile addressing is the default on new 11i v3 installations, but the I/O subsystem still recognizes the pre-11.i v3 nomenclature. This means that you are not required to convert to agile addressing when you upgrade to 11i v3, though you should seriously consider its advantages.

For instructions on migrating a system to agile addressing, see the white paper *LVM Migration from Legacy to Agile Naming Model* at http://www.hp.com/go/hpux-core-docs.

NOTE: It is possible, though not a best practice, to use legacy DSFs (that is, DSFs using the older naming convention) on some nodes after migrating to agile addressing on others; this allows you to migrate different nodes at different times, if necessary. For information on migrating cluster lock volumes to agile addressing, see "Updating the Cluster Lock Configuration" (page 281).

For more information about agile addressing, see following documents in the `11i v3` collection at http://www.hp.com/go/hpux-core-docs:

- the *Logical Volume Management* volume of the HP-UX *System Administrator's Guide*
- the *HP-UX 11i v3 Installation and Update Guide*
- the white papers:
  - *Overview: The Next Generation Mass Storage Stack*
  - *HP-UX 11i v3 Persistent DSF Migration Guide*
  - *LVM Migration from Legacy to Agile Naming Model*
  - *HP-UX 11i v3 Native Multi-Pathing for Mass Storage*

See also the HP-UX 11i v3 `intro(7)` manpage, and"About Multipathing" (page 32).

NOTE: As of A.11.20, Serviceguard supports cluster-wide DSFs, and HP recommends that you use them. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

Examples of Mirrored Storage

Figure 29 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`.

NOTE: Under agile addressing (see "About Device File Names (Device Special Files)" (page 77)), the storage units in this example would have names such as `disk1`, `disk2`, `disk3`, etc.

As of A.11.20, Serviceguard supports cluster-wide DSFs, and HP recommends that you use them. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

Figure 29 Physical Disks Within Shared Storage Units

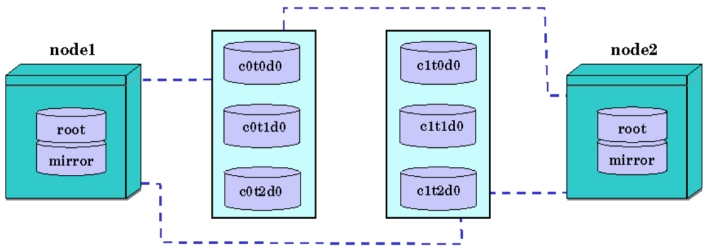


Figure 30 shows the individual disks combined in a multiple disk mirrored configuration.

Figure 30 Mirrored Physical Disks

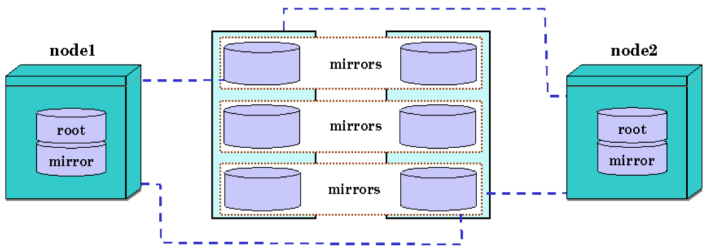
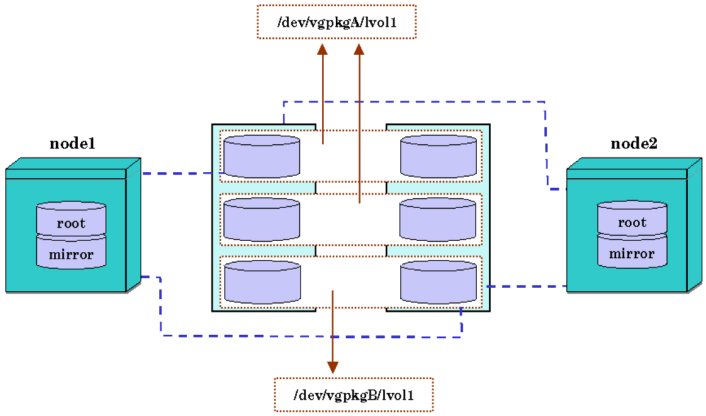


Figure 31 shows the mirrors configured into LVM volume groups, shown in the figure as `/dev/vgpkgA` and `/dev/vgpkgB`. The volume groups are activated by Serviceguard packages for use by highly available applications.

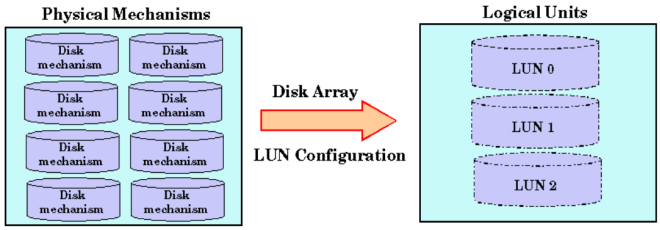
Figure 31 Multiple Devices Configured in Volume Groups



Examples of Storage on Disk Arrays

Figure 32 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 32 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.

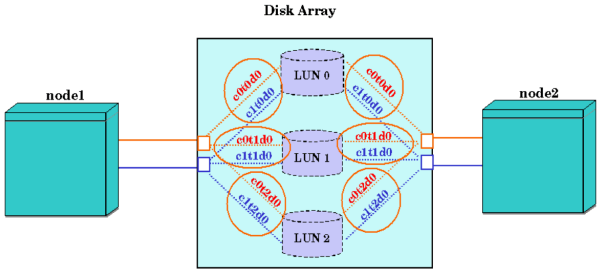
Figure 33 shows LUNs configured with multiple paths (links) to provide redundant pathways to the data.

NOTE: Under agile addressing, the storage units in these examples would have names such as `disk1`, `disk2`, `disk3`, etc. See "About Device File Names (Device Special Files)" (page 77)

As of A.11.20, Serviceguard supports cluster-wide DSFs, and HP recommends that you use them. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

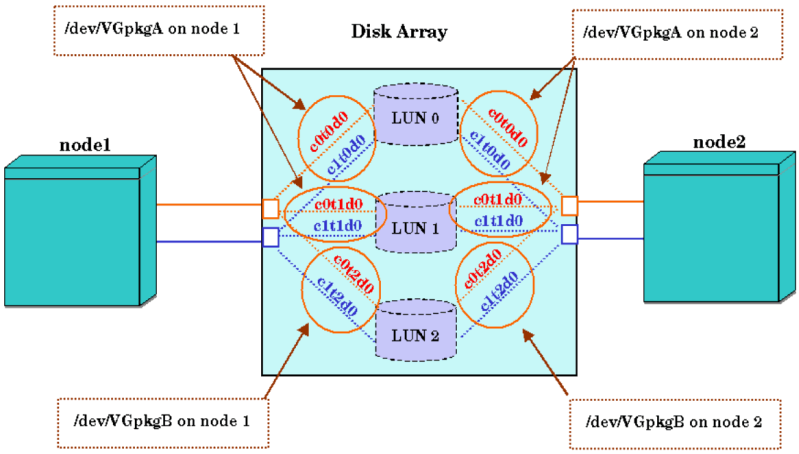
.

Figure 33 Multiple Paths to LUNs



Finally, the multiple paths are configured into volume groups as shown in Figure 34.

Figure 34 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

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 the HP-UX *System Administrator's Guide*. Release Notes for Veritas Volume Manager contain a description of Veritas volume management products.

HP-UX Logical Volume Manager (LVM)

Logical Volume Manager (LVM) is the default 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. 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 `VEA`. 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 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.

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. (required with CFS)
- do not have all nodes cabled to all disks. (required with CFS)
- need to use software RAID mirroring or striped mirroring.
- have multiple heartbeat subnets configured.

Propagation of Disk Groups in VxVM

A VxVM disk group can be created on any node, whether the cluster is up or not. You must validate the disk group by trying to import it on each 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)

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information on CVM support: http://www.hp.com/go/hpux-serviceguard-docs.

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, 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 VxVM Full Product and CVM are enhanced versions of the VxVM volume manager specifically designed for cluster use. When installed with the Veritas Volume Manager, 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, which runs on all nodes in the cluster. The cluster must be up and must be running this package before you can configure VxVM disk groups for use with CVM. Disk groups must be created from the CVM Master node. The Veritas CVM package for CVM version 4.1 and later is named `SG-CFS-pkg`.

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 supports concurrent storage read/write access between multiple nodes by applications which can manage read/write access contention, such as Oracle Real Application Cluster (RAC).

CVM 4.1 and later can be used with Veritas Cluster File System (CFS) in Serviceguard. Several of the HP Serviceguard Storage Management Suite bundles include features to enable both CVM and CFS.

CVM can be used in clusters that:

- run applications that require fast disk group activation after package failover;
- require storage 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;
- run applications, such as Oracle RAC, that require concurrent storage read/write access between multiple nodes.

For heartbeat requirements, see "Redundant Heartbeat Subnets" (page 83).

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

All shared disk groups (DGs) are imported when the system multi-node's control script starts up CVM. 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 system multi-node package 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

CVM disk groups are created on one cluster node known as the CVM master node. CVM verifies that each node can see each disk and will not allow invalid DGs to be created.

Redundant Heartbeat Subnets

HP recommends that you configure all subnets that connect cluster nodes as heartbeat networks; this increases protection against multiple faults at no additional cost.

For CVM 4.1 and later, the minimum recommended configurations are:

1) dual (multiple) heartbeat networks

2) single heartbeat network with standby LAN card(s)

Comparison of Volume Managers

The following table summarizes the advantages and disadvantages of the volume managers.

Table 5 Pros and Cons of Volume Managers with Serviceguard

Product	Advantages	Tradeoffs
Logical Volume Manager (LVM)	• Software is provided with all versions of HP-UX. • Provides up to 3-way mirroring using optional Mirrordisk/UX software. • Dynamic multipathing (DMP) is active by default as of HP-UX 11i v3. • Supports exclusive activation as well as read-only activation from multiple nodes • Can be used to configure a cluster lock disk • Supports multiple heartbeat subnets; the one with the faster failover time is used to re-form the cluster.	• Lacks flexibility and extended features of some other volume managers
Mirrordisk/UX	• Software mirroring • Lower cost solution	• Lacks extended features of other volume managers
Shared Logical Volume Manager (SLVM)	• Provided free with SGeRAC for multi-node access to RAC data • Supports up to 16 nodes in shared read/write mode for each cluster • Supports exclusive activation • Supports multiple heartbeat subnets. • Online node configuration with activated shared volume groups (using specific SLVM kernel and Serviceguard revisions)	• Lacks the flexibility and extended features of some other volume managers. • Limited mirroring support
Base-VxVM	• Software is supplied free with HP-UX 11i releases. • Java-based administration through graphical user interface. • Striping (RAID 0) support. • Concatenation. • Online resizing of volumes. • Supports multiple heartbeat subnets.	• Cannot be used for a cluster lock • Supports only exclusive read or write activation • Package delays are possible, due to lengthy `vxdg` import at the time the package is started or failed over
Veritas Volume Manager— Full VxVM product	• Disk group configuration from any node. • DMP for active/active storage devices. • Supports exclusive activation. • Hot relocation and unrelocation of failed subdisks • Supports up to 32 plexes per volume • RAID 1+0 mirrored stripes • RAID 1 mirroring • RAID 5 • RAID 0+1 striped mirrors • Supports multiple heartbeat subnets, which could reduce cluster reformation time.	• Requires purchase of additional license • Cannot be used for a cluster lock • Does not support activation on multiple nodes in either shared mode or read-only mode • May cause delay at package startup time due to lengthy `vxdg` import
Veritas Cluster Volume Manager	• Provides volume configuration propagation. • Supports cluster shareable disk groups. • Package startup time is faster than with VxVM.	• Disk groups must be configured on a master node • Cluster startup may be slower than with VxVM

Table 5 Pros and Cons of Volume Managers with Serviceguard *(continued)*

Product	Advantages	Tradeoffs
	• Supports shared activation. • Supports exclusive activation. • Supports activation in different modes on different nodes at the same time • RAID 1+0 mirrored stripes • RAID 0+1 striped mirrors • CVM versions 4.1 and later support the Veritas Cluster File System (CFS)	• Requires purchase of additional license • No support for RAID 5 • CVM requires all nodes to have connectivity to the shared disk groups • Not currently supported on all versions of HP-UX

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.

System Reset When a Node Fails

The most dramatic response to a failure in a Serviceguard cluster is an HP-UX TOC or INIT, which is a system reset without a graceful shutdown (normally referred to in this manual simply as a system reset). This allows packages to move quickly to another node, protecting the integrity of the data.

A system reset occurs if a cluster node cannot communicate with the majority of cluster members for the predetermined time, or under other circumstances such as a kernel hang or failure of the cluster daemon (`cmcld`).

The case is covered in more detail under "What Happens when a Node Times Out" (page 85). See also "Cluster Daemon: cmcld" (page 39).

A system reset is also initiated by Serviceguard itself under specific circumstances; see "Responses to Package and Service Failures " (page 87).

What Happens when a Node Times Out

Each node sends a heartbeat message to all other nodes at an interval equal to one-fourth of the value of the configured *`MEMBER_TIMEOUT`* or 1 second, whichever is less. You configure *`MEMBER_TIMEOUT`* in the cluster configuration file (see "Cluster Configuration Parameters " (page 105)); the heartbeat interval is not directly configurable. If a node fails to send a heartbeat message within the time set by *`MEMBER_TIMEOUT`*, the cluster is reformed minus the node no longer sending heartbeat messages.

When a node detects that another node has failed (that is, no heartbeat message has arrived within *`MEMBER_TIMEOUT`* microseconds), the following sequence of events occurs:

1. The node contacts the other nodes and tries to re-form the cluster without the failed node.
2. If the remaining nodes are a majority or can obtain the cluster lock, they form a new cluster without the failed node.
3. If the remaining nodes are not a majority or cannot get the cluster lock, they halt (system reset).

Example

Situation. Assume a two-node cluster, with `Package1` running on `SystemA` and `Package2` running on `SystemB`. Volume group `vg01` is exclusively activated on `SystemA`; volume group `vg02`is exclusively activated on `SystemB`. Package IP addresses are assigned to `SystemA` and `SystemB` respectively.

Failure. Only one LAN has been configured for both heartbeat and data traffic. During the course of operations, heavy application traffic monopolizes the bandwidth of the network, preventing heartbeat packets from getting through.

Since `SystemA` does not receive heartbeat messages from `SystemB`, `SystemA` attempts to reform as a one-node cluster. Likewise, since `SystemB` does not receive heartbeat messages from `SystemA`, `SystemB` also attempts to reform as a one-node cluster. During the election protocol, each node votes for itself, giving both nodes 50 percent of the vote. Because both nodes have 50 percent of the vote, both nodes now vie for the cluster lock. Only one node will get the lock.

Outcome. Assume `SystemA` gets the cluster lock. `SystemA` reforms as a one-node cluster. After re-formation, `SystemA` will make sure all applications configured to run on an existing cluster node are running. When `SystemA` discovers `Package2` is not running in the cluster it will try to start `Package2` if `Package2` is configured to run on `SystemA`.

`SystemB` recognizes that it has failed to get the cluster lock and so cannot re-form the cluster. To release all resources related to `Package2` (such as exclusive access to volume group `vg02` and the `Package2` IP address) as quickly as possible, `SystemB` halts (system reset).

NOTE: If *`AUTOSTART_CMCLD`* in `/etc/rc.config.d/cmcluster` (`$SGAUTOSTART`) is set to zero, the node will not attempt to join the cluster when it comes back up.

For more information on cluster failover, see the white paper *Optimizing Failover Time in a Serviceguard Environment (version A.11.19 and later)* at http://www.hp.com/go/hpux-serviceguard-docs. For troubleshooting information, see "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320).

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 failover packages currently running on that node to an adoptive node elsewhere in the cluster. (System multi-node and multi-node packages do not fail over.)

The new location for each failover 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 swiftly 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 system reset. If a monitored data LAN interface fails without a standby, the node fails with a system reset only if *`node_fail_fast_enabled`* (page 224) is set to `YES` for the package. Otherwise any packages using that LAN interface will be halted and moved to another node if possible (unless the LAN recovers immediately; see "When a Service, Subnet, or Monitored Resource Fails, or a Dependency is Not Met" (page 61)).

Disk protection is provided by separate products, such as Mirrordisk/UX in LVM or Veritas mirroring in VxVM and related products. 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 Availability Monitors*, which you can find at the address given in the preface to this manual.

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).

Responses to Package and Service Failures

In the default case, the failure of a failover package, or of a service within the 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. A package will also fail if it is configured to have a dependency on another package, and that package fails. If the package manager receives a report of an EMS (Event Monitoring Service) event showing that a configured resource dependency is not met, the package fails and tries to restart on the alternate node.

You can modify this default behavior by specifying that the node should halt (system reset) before the transfer takes place. You do this by setting failfast parameters in the package configuration file.

In cases where package shutdown might hang, leaving the node in an unknown state, failfast options can provide a quick failover, after which the node will be cleaned up on reboot. Remember, however, that a system reset causes *all* packages on the node to halt abruptly without a clean shutdown.

The settings of the failfast parameters in the package configuration file determine the behavior of the package and the node in the event of a package or resource failure:

- If *service_fail_fast_enabled*is set to `yes` in the package configuration file, Serviceguard will halt the node with a system reset if there is a failure of that specific service.
- If *node_fail_fast_enabled* is set to `yes` in the package configuration file, and the package fails, Serviceguard will halt (system reset) the node on which the package is running.

NOTE: In a very few cases, Serviceguard will attempt to reboot the system before a system reset when this behavior is specified. If there is enough time to flush the buffers in the buffer cache, the reboot succeeds, and a system reset does not take place. Either way, the system will be guaranteed to come down within a predetermined number of seconds.

"Choosing Switching and Failover Behavior" (page 127) provides advice on choosing 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; see "What Happens when a Node Times Out" (page 85).

4 Planning and Documenting an HA Cluster

Building a Serviceguard cluster begins with a planning phase in which you gather information about all the hardware and software components of the configuration.

This chapter assists you in the following planning areas:

- General Planning
- Hardware Planning (page 89)
- Power Supply Planning (page 93)
- Using a Quorum Server (page 95)
- LVM Planning (page 96)
- CVM and VxVM Planning (page 98)
- Cluster Configuration Planning (page 99)
- Package Configuration Planning (page 120)

Blank Planning Worksheets (page 355) contains a set of blank worksheets which you may find useful as an offline record of important details of the configuration.

NOTE: Planning and installation overlap considerably, so you may not be able to complete the worksheets before you proceed to the actual configuration. In that case, 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 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

Serviceguard requires approximately 15.5 MB of lockable memory.

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:

- 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 (Event Monitoring Service) 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.

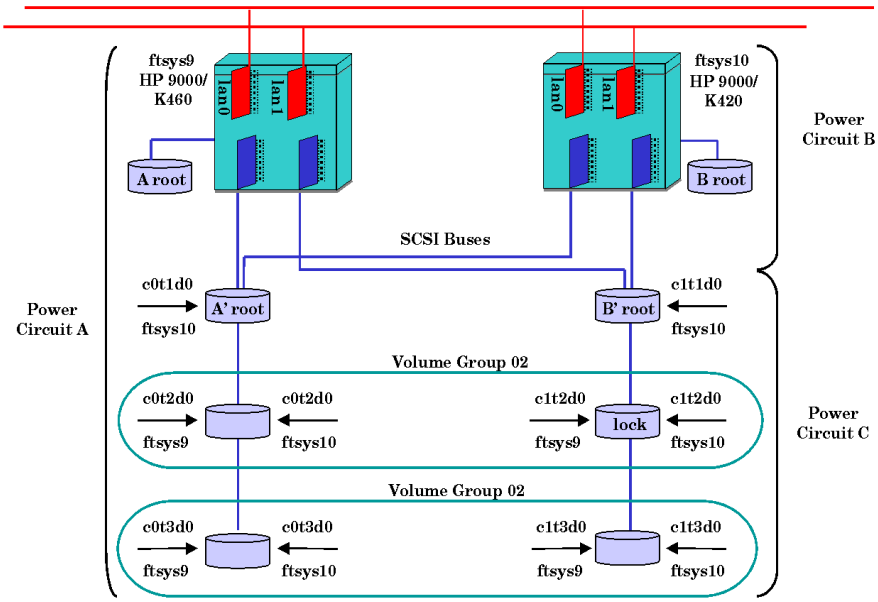
Hardware Planning

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 35.

NOTE: Under agile addressing, the storage units in this example would have names such as `disk1`, `disk2`, `disk3`, etc. See "About Device File Names (Device Special Files)" (page 77).

As of A.11.20, Serviceguard supports cluster-wide DSFs, and HP recommends that you use them. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

Figure 35 Sample Cluster Configuration



Create a similar sketch for your own cluster. You may also find it useful to record the information as in the sample Hardware Worksheet (page 93), indicating which device adapters occupy which

slots, and the bus address for each adapter; you can update the details as you do the cluster configuration (described in Chapter 5).

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. HP-UX workstations are not supported for Serviceguard.

Collect information for the following items; see the Hardware Configuration Worksheet (page 93) for an example:

Server Number	The series number; for example, `rp8400 or rx8620-32`.
Host Name	The name to be used on the system as the host name.
Memory Capacity	The memory in MB.
Number of I/O slots	The number of slots.

Network Information

Serviceguard monitors LAN interfaces.

NOTE: Serviceguard supports communication across routers between nodes in the same cluster; for more information, see the documents listed under "Cross-Subnet Configurations" (page 29).

Serviceguard communication relies on the exchange of DLPI (Data Link Provider Interface) traffic (for nodes on the same subnet) and UDP/TCP (User Datagram Protocol/Transmission Control Protocol) and ICMP traffic 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.

NOTE: In a cross-subnet configuration, in which the cluster spans subnets joined by a router, a standby for each subnet is required. See "Cross-Subnet Configurations" (page 29) for more information.

HP recommends that you configure heartbeats on all subnets, including those to be used for client data.

Collect the following information for each LAN interface; see the Hardware Configuration Worksheet (page 93) for an example:

Subnet Name	The IP address mask for the subnet. Note that, except in cross-subnet configurations (page 29), heartbeat IP addresses must be on the same subnet on each node.
Interface Name	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	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 H (page 364).

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

- INOUT
- INONLY_OR_INOUT

The default is INOUT.

See "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66) for more information.

NETWORK_AUTO_FAILBACK

See the *NETWORK_AUTO_FAILBACK* parameter description under"Cluster Configuration Parameters " (page 105).

Kind of LAN Traffic

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

- Heartbeat
- Client Traffic
- Standby

This information is used in creating the subnet groupings and identifying the IP addresses used in the cluster and package configuration files.

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 6 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

Table 6 SCSI Addressing in Cluster Configuration *(continued)*

System or Disk	Host Interface SCSI Address
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

Collect the following information for each disk connected to each disk device adapter on the node:

Bus Type — Indicate the type of bus. Supported busses are Fibre Channel and 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 (for legacy DSFs) or ioscan -fnNC disk (for agile addressing).

This information 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` or ioscan -fnNC disk
- `lssf /dev/*dsk/*`
- `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 complete information about 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. You can mark up a printed listing of the output from the `lssf` command to indicate which physical volume group a disk should be assigned to.

Hardware Configuration Worksheet

You may find a worksheet such as the following useful for organizing and recording your cluster hardware configuration. This worksheet is an example; blank worksheets are in Appendix E. Make as many copies as you need.

```
    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


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

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

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

    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 a device has redundant power supplies, connect each power supply to a separate power circuit. 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 each device in a cluster has three power supplies, you will need a minimum of three separate power circuits to eliminate electrical power as a single point of failure for the cluster.

In the case of hardware with only one power supply, no more than half the nodes should share a single power circuit. If a power source supplies exactly half the nodes, it must not also supply the cluster disk lock or Quorum Server, or the cluster will not be able to re-form 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 disk (if any). If you use a Quorum Server, or quorum server cluster, make sure each quorum server node has a power source separate from that of every cluster it serves. If you use software mirroring, make sure power supplies are not shared among different physical volume groups (or VxVM disk groups); this allows you to set up mirroring between physical disks that are not only on different I/O buses, but also connected to different power supplies.

To prevent confusion, label each hardware unit and power supply unit clearly with a different unit number. You can use a Power Supply Worksheet such as the one that follows to record the hardware

units you are using and the power supply to which they will be connected; record the following information:

Host Name	The host name for each SPU.
Disk Unit	The disk drive unit number for each disk.
Tape Unit	The tape unit number for each backup device.
Other Unit	Tthe number of any other unit.
Power Supply	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

You may find the following worksheet useful to help you organize and record your power supply configuration. This worksheet is an example; blank worksheets are in Appendix E. Make as many copies as you need.

```
=====================================================================
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 _____________________
```

Cluster Lock Planning

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 an LVM lock disk, a lock LUN, or a Quorum Server as the cluster lock. For more information about the cluster lock, and requirements and recommendations, see "Cluster Lock " (page 44).

NOTE: You cannot use more than one type of lock in the same cluster.

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

Quorum Server as the cluster lock. In selecting a cluster lock configuration, be careful to anticipate any potential need for additional cluster nodes.

For more information on lock disks, lock LUNs, and the Quorum Server, see "Choosing Cluster Lock Disks" (page 164), "Setting Up a Lock LUN" (page 165), and "Setting Up and Running the Quorum Server" (page 168).

Cluster Lock Disk and Re-formation Time

`cmquerycl (1m)` displays the failover time for the cluster in the `Possible Cluster Lock Devices` section, for example:

```
Possible Cluster Lock Devices:

/dev/dsk/c0t1d4                          30 seconds

Note: all lock disks have the same failover time
```

All disks in all volume groups which are accessible by all nodes in the cluster are potential cluster lock devices, but because it takes a fixed amount of time to acquire the cluster lock, regardless of the lock device, only the first potential lock device is displayed. You may need to choose a different device because of power considerations; see "Power Supply Planning " (page 93).

Planning for Expansion

Bear in mind that a cluster with more than 4 nodes cannot use a lock disk or lock LUN. Thus, if you plan to add enough nodes to bring the total to more than 4, you should use a Quorum Server.

Using a Quorum Server

The operation of Quorum Server is described under "Use of the Quorum Server as the Cluster Lock" (page 46). See also "Cluster Lock " (page 44), "Specifying a Quorum Server" (page 180), and the *`QS_HOST`* and *`QS_ADDR`* parameter descriptions under "Cluster Configuration Parameters " (page 105).

IMPORTANT: If you plan to use a Quorum Server, make sure you read the *HP Serviceguard Quorum Server Version A.04.00 Release Notes* before you proceed. You can find them at: http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard Quorum Server Software` . You should also consult the Quorum Server white papers at the same location.

A quorum server:

- Can be used with up to 150 clusters, not exceeding 300 nodes total.
- Can support a cluster with any supported number of nodes.
- Can communicate with the cluster on up to two subnets (a primary and an alternate) using either IPv4 or IPv6 addresses.

Quorum Server Worksheet

You may find it useful to use a Quorum Server worksheet, as in the example that follows, to identify a quorum server for use with one or more clusters. You may also want to enter quorum server host and timing parameters on the Cluster Configuration Worksheet. Blank worksheets are in Blank Planning Worksheets (page 355).

You can use the Quorum Server worksheet to record the following:

`Quorum Server Host`	The host name (and alternate address, if any) for the quorum server.
`IP Address`	The IP address(es) by which the quorum server will be connected to the cluster.

Supported Node Names The name (39 characters or fewer) of each cluster node 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 Data:
==============================================================================

    QS Hostname: _____________IP Address: _________________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 as 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 group 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 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.
- HP recommends that you use volume group names other than the default volume group names (`vg01`, `vg02`, etc.). Choosing volume group names that represent the high availability applications that they are associated with (for example, `/dev/vgdatabase` will simplify cluster administration).
- Logical Volume Manager (LVM) 2.0 volume groups, which remove some of the limitations imposed by LVM 1.0 volume groups, can be used on systems running some recent versions of HP-UX 11i v3 and Serviceguard. Check the Release Notes for your version of Servicegaurd for details. For more information, see the white paper *LVM 2.0 Volume Groups in HP-UX 11i v3* at http://www.hp.com/go/hpux-core-docs -> HP–UX 11i v3.

Using EMS to Monitor Volume Groups

You can use EMS (Event Monitoring Service) resource monitors to monitor the status of LVM volume groups used by packages. You do this by defining a resource for the package, as in the example that follows.

NOTE: EMS cannot be used to monitor the status of VxVM disk groups. For this you should use the volume monitor `cmvolmond` which is supplied with Serviceguard. `cmvolmond` can also monitor LVM volumes. See "About the Volume Monitor" (page 123).

```
resource_name /vg/vgpkg/pv_summary
resource_polling_interval 60
resource_start AUTOMATIC
resource_up_value = UP
```

The example above will monitor all PV links for the volume group `vgpkg`. As long as all devices within the `vgpkg` volume group are functional, the resource will remain in `UP` status. When the last path to the storage for the volume group fails, or any device within the volume group fails, the resource value will change, and at the next polling interval the package will fail because the resource no longer meets the package requirements. The package can then fail over to any other node for which this resource is still in the `UP` status.

IMPORTANT: You *must* set the IO timeout for all logical volumes within the volume group being monitored to something other than the default of zero (no timeout); otherwise the EMS resource monitor value will never change upon a failure. Suggested IO timeout values are 20 to 60 seconds. See "Setting Logical Volume Timeouts" (page 171) for more information.

For more information, see "Using the EMS HA Monitors" (page 56).

LVM Worksheet

You may find a worksheet such as the following useful to help you organize and record your physical disk configuration. This worksheet is an example; blank worksheets are in Appendix E (page 355). Make as many copies as you need.

NOTE: Under agile addressing, the physical volumes in the sample worksheet that follows would have names such as `disk1`, `disk2`, etc. See "About Device File Names (Device Special Files)" (page 77).

As of A.11.20, Serviceguard supports cluster-wide DSFs, and HP recommends that you use them. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

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

    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.

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information on CVM support: http://www.hp.com/go/hpux-serviceguard-docs.

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

- 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 that is *not* shared among cluster nodes.
- A cluster lock disk must be configured into an LVM volume group; you cannot use a VxVM or CVM disk group. (See "Cluster Lock Planning" (page 94) for information about cluster lock options.)
- VxVM disk group names should not be entered into the cluster configuration file. These names are not inserted into the cluster configuration file by `cmquerycl`.

CVM and VxVM Worksheet

You may find a worksheet such as the following useful to help you organize and record your specific physical disk configuration. This worksheet is an example; blank worksheets are in Appendix E (page 355).

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

Disk Group Name: ______dg01_____________________
  Disk Name: ____________________c1t2d0_____________________________
  Disk Name: ____________________c2t2d0_____________________________
  Disk Name: ____________________c3t2d0_____________________________
  Disk Name: ____________________________________________________
CFS DG Package Name: ___________SG-CFS-DG_1_____
CFS Volume, MP, and MP Pkg: _logdata_/mnt/lvol1_ SG-CFS-MP_1_______
CFS Volume, MP, and MP Pkg: ____________________________________________

Disk Group Name: ______dg02_____________________
Disk Name: ____________________c1t3d0_____________________________
Disk Name: ____________________c2t3d0_____________________________
Disk Name: ____________________c3t3d0_____________________________
CFS DG Package Name: ___________SG-CFS-DG_2_____
CFS Volume, MP, and MP Pkg: _hrdata__/mnt/lvol4_ SG-CFS-MP_2____
CFS Volume, MP, and MP Pkg: ____________________________________________
```

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 value of the cluster *MEMBER_TIMEOUT*.

  See *MEMBER_TIMEOUT* under "Cluster Configuration Parameters " (page 105) for recommendations.

- 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 files in the `/usr` directory, must be the same on all nodes.

About Cluster-wide Device Special Files (cDSFs)

Under agile addressing on HP-UX 11i v3, each device has a unique identifier as seen from a given host; this identifier is reflected in the name of the Device Special File (DSF). See "About Device File Names (Device Special Files)" (page 77) for more information.

Because DSF names may be duplicated between one host and other, it is possible for different storage devices to have the same name on different nodes in a cluster, and for the same piece of storage to be addressed by different names. Cluster-wide device files (cDSFs), available as of the September 2010 HP-UX Fusion Release, ensure that each storage device used by the cluster has a unique device file name.

IMPORTANT: Check the latest version of the release notes (at the address given in the preface to this manual) for information about Serviceguard support for cDSFs.

HP recommends that you use cDSFs for the storage devices in the cluster because this makes it simpler to deploy and maintain a cluster, and removes a potential source of configuration errors. See "Creating Cluster-wide Device Special Files (cDSFs)" (page 150) for instructions.

Points To Note

- cDSFs can be created for any group of nodes that you specify, provided that Serviceguard A.11.20 is installed on each node.

  Normally, the group should comprise the entire cluster.

- cDSFs apply only to shared storage; they will not be generated for local storage, such as root, boot, and swap devices.
- Once you have created cDSFs for the cluster, HP-UX automatically creates new cDSFs when you add shared storage.
- HP recommends that you do not mix cDSFs with persistent (or legacy) DSFs in a volume group. You cannot use `cmpreparestg (1m)` on a volume group in which they are mixed.

  See "About Easy Deployment" (page 100) for more information about `cmpreparestg`.

Where cDSFs Reside

cDSFs reside in two new HP-UX directories, `/dev/cdisk` for cluster-wide block devicefiles and `/dev/rcdisk` for cluster-wide character devicefiles. Persistent DSFs that are not cDSFs continue to reside in `/dev/disk` and `/dev/rdisk`, and legacy DSFs (DSFs using the naming convention that was standard before HP–UX 11i v3) in `/dev/dsk` and `/dev/rdsk`. It is possible that a storage device on an 11i v3 system could be addressed by DSFs of all three types of device — but if you are using cDSFs, you should ensure that you use them exclusively as far as possible.

NOTE: Software that assumes DSFs reside only in `/dev/disk` and `/dev/rdisk` will not find cDSFs and may not work properly as a result; as of the date of this manual, this was true of the Veritas Volume Manager, VxVM.

Limitations of cDSFs

- cDSFs are supported only within a single cluster; you cannot define a cDSF group that crosses cluster boundaries.
- A node can belong to only one cDSF group.
- cDSFs are not supported by VxVM, CVM, CFS, or any other application that assumes DSFs reside only in `/dev/disk` and `/dev/rdisk`.
- cDSFs do not support disk partitions.

  Such partitions can be addressed by a device file using the agile addressing scheme, but not by a cDSF.

LVM Commands and cDSFs

Some HP-UX commands have new options and behavior to support cDSFs, specifically:

- `vgimport -C` causes `vgimport (1m)` to use cDSFs.
- `vgscan -C` causes `vgscan (1m)` to display cDSFs

See the manpages for more information.

The following new HP-UX commands handle cDSFs specifically:

- `vgcdsf(1m)` converts all persistent DSFs in a volume group to cDSFs.

  Legacy DSFs in the volume group will not be converted, but you can use HP-UX the `vgdsf` script to convert these legacy DSFs to persistent DSFs if you need to. For more information on the `vgdsf` script, see the white paper *LVM Migration from Legacy to Agile Naming Model* at http://http://www.hp.com/go/hpux-core-docs. For more information on `vgcdsf`, see the manpage.

- `io_cdsf_config (1m)` displays information about cDSFs.

  See the manpage for more information.

About Easy Deployment

In the past you had two main choices for configuring a cluster: using Serviceguard commands, as described in detail in Chapter 5 (page 149), or using the Serviceguard Manager GUI (or some combination of these two methods). As of Serviceguard A.11.20, there is a third option, called Easy Deployment.

The Easy Deployment tool consists of three commands: `cmpreparecl (1m)`, `cmdeploycl (1m)`, and `cmpreparestg (1m)`. These commands allow you to get a cluster up and running in the minimum amount of time. The commands:

- Configure networking and security (`cmpreparecl`)
- Create and start the cluster with a cluster lock device (`cmdeploycl`)
- Create or modify Logical volume groups and VxVM/CVM disk groups and import volume groups or disk groups as additional shared storage for use by cluster packages (`cmpreparestg`)

Advantages of Easy Deployment

- Quick and simple way to create and start a cluster.
- Automates security and networking configuration that must always be done before you configure nodes into a cluster.
- Simplifies cluster lock configuration.
- Simplifies creation of shared storage for packages.

Limitations of Easy Deployment

- Does not install or verify Serviceguard software
- Requires agile addressing for disks. See "About Device File Names (Device Special Files)" (page 77).
- `cmpreparestg (1m)` will fail if cDSFs and persistent DSFs are mixed in a volume group. See "About Cluster-wide Device Special Files (cDSFs)" (page 99) for more information about cDSFs.
- Does not configure access control policies.
- Does not install or configure firewall and related software.
- Does not support cross-subnet configurations.
- Does not configure packages.
- Does not discover or configure a quorum server (but can deploy one that is already configured).
- Does not support asymmetric network configurations (in which only a subset of nodes has access to a given subnet).

For more information and instructions, see "Using Easy Deployment" (page 152).

Heartbeat Subnet and Cluster Re-formation Time

The speed of cluster re-formation depends on the number of heartbeat subnets.

If the cluster has only a single heartbeat network, and a network card on that network fails, heartbeats will be lost while the failure is being detected and the IP address is being switched to a standby interface. The cluster may treat these lost heartbeats as a failure and re-form without one or more nodes. To prevent this, a minimum *`MEMBER_TIMEOUT`* value of 14 seconds is required for clusters with a single heartbeat network.

If there is more than one heartbeat subnet, and there is a failure on one of them, heartbeats will go through another, so you can configure a smaller *`MEMBER_TIMEOUT`* value.

NOTE: For heartbeat configuration requirements, see the discussion of the *HEARTBEAT_IP* parameter later in this chapter. For more information about managing the speed of cluster re-formation, see the discussion of the *MEMBER_TIMEOUT* parameter, and further discussion under "What Happens when a Node Times Out" (page 85) ,"Modifying the MEMBER_TIMEOUT Parameter" (page 183), and, for troubleshooting, "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320).

About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode

As of A.11.19, Serviceguard supports three possibilities for resolving the nodes' hostnames (and Quorum Server hostnames, if any) to network address families:

- IPv4-only
- IPv6-only
- Mixed

IPv4-only means that Serviceguard will try to resolve the hostnames to IPv4 addresses only.

IMPORTANT: You can configure an IPv6 heartbeat, or stationary or relocatable IP address, in any mode: IPv4-only, IPv6-only, or mixed. You can configure an IPv4 heartbeat, or stationary or relocatable IP address, in IPv4-only or mixed mode.

IPv6-only means that Serviceguard will try to resolve the hostnames to IPv6 addresses only.

Mixed means that when resolving the hostnames, Serviceguard will try both IPv4 and IPv6 address families.

You specify the address family the cluster will use in the cluster configuration file (by setting *HOSTNAME_ADDRESS_FAMILY* to `IPV4`, `IPV6`, or `ANY`), or by means of the `-a` option of `cmquerycl (1m)`; see "Specifying the Address Family for the Cluster Hostnames" (page 178). The default is `IPV4`. See the subsections that follow for more information and important rules and restrictions.

What Is IPv4–only Mode?

IPv4 is the default mode: unless you specify `IPV6` or `ANY` (either in the cluster configuration file or via `cmquerycl -a`) Serviceguard will always try to resolve the nodes' hostnames (and the Quorum Server's, if any) to IPv4 addresses, and will not try to resolve them to IPv6 addresses. This means that you must ensure that each hostname can be resolved to at least one IPv4 address.

NOTE: This applies only to hostname resolution. You can have IPv6 heartbeat and data LANs no matter what the *HOSTNAME_ADDRESS_FAMILY* parameter is set to. (IPv4 heartbeat and data LANs are allowed in IPv4 and mixed mode.)

What Is IPv6-Only Mode?

If you configure IPv6-only mode (*HOSTNAME_ADDRESS_FAMILY* set to `IPV6`, or **`cmquerycl -a ipv6`**), then all the hostnames and addresses used by the cluster — including the heartbeat and stationary and relocatable IP addresses, and Quorum Server addresses if any — must be or resolve to IPv6 addresses. The single exception to this is each node's IPv4 loopback address, which cannot be removed from `/etc/hosts`.

NOTE: How the *clients* of IPv6-only cluster applications handle hostname resolution is a matter for the discretion of the system or network administrator; there are no HP requirements or recommendations specific to this case.

In IPv6-only mode, all Serviceguard daemons will normally use IPv6 addresses for communication among the nodes, although local (intra-node) communication may occur on the IPv4 loopback address.

For more information about IPv6, see Appendix H (page 364).

Rules and Restrictions for IPv6-Only Mode

IMPORTANT: See the latest version of the Serviceguard release notes for the most current information on these and other restrictions.

- All addresses used by the cluster must be in each node's `/etc/hosts` file. In addition, the file must contain the following entry:

  ```
  ::1 localhost ipv6-localhost ipv6-loopback
  ```

  For more information and recommendations about hostname resolution, see "Configuring Name Resolution" (page 159).
- All addresses must be IPv6, apart from the node's IPv4 loopback address, which cannot be removed from `/etc/hosts`.
- The node's public LAN address (by which it is known to the outside world) must be the last address listed in `/etc/hosts`.

  Otherwise there is a possibility of the address being used even when it is not configured into the cluster.
- You must use `$SGCONF/cmclnodelist`, not `~/.rhosts` or `/etc/hosts.equiv`, to provide root access to an unconfigured node.

  NOTE: This also applies if *`HOSTNAME_ADDRESS_FAMILY`* is set to `ANY`. See "Allowing Root Access to an Unconfigured Node" (page 157) for more information.
- If you use a Quorum Server, you must make sure that the Quorum Server hostname (and the alternate Quorum Server address specified by `QS_ADDR`, if any) resolve to IPv6 addresses, and you must use Quorum Server version A.04.00 or later. See the latest Quorum Server release notes for more information; you can find them at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard Quorum Server Software`.

  NOTE: The Quorum Server itself can be an IPv6–only system; in that case it can serve IPv6–only and mixed-mode clusters, but not IPv4–only clusters.
- If you use a Quorum Server, and the Quorum Server is on a different subnet from cluster, you must use an IPv6-capable router.
- Hostname aliases are not supported for IPv6 addresses, because of operating system limitations.

  NOTE: This applies to all IPv6 addresses, whether *`HOSTNAME_ADDRESS_FAMILY`* is set to `IPV6` or `ANY`.
- Cross-subnet configurations are not supported in IPv6-only mode.

  NOTE: This also applies if *`HOSTNAME_ADDRESS_FAMILY`* is set to `ANY`. See "Cross-Subnet Configurations" (page 29) for more information about such configurations.
- VxVM, VxFS, and CVM/CFS are supported in IPv6–only mode on HP Serviceguard A.11.20 April 2011 patches with Veritas Storage Foundation™ 5.1 SP1 and later.

NOTE: CVM/CFS can be *installed* on any system in an IPv6–only cluster using HP Serviceguard A.11.20 April 2011 patches with Veritas Storage Foundation™ 5.1 SP1 and later, even if they are not configured.

This also means that the Serviceguard component of bundles that include Serviceguard cannot be configured in IPV6 mode prior to Veritas Storage Foundation™ 5.1 SP1.

- HPVM is not supported IPv6-only mode. You cannot configure a virtual machine either as a node or a package in an IPv6-only cluster.

Recommendations for IPv6-Only Mode

IMPORTANT: Check the current Serviceguard release notes for the latest instructions and recommendations.

- If you decide to migrate the cluster to IPv6-only mode, you should plan to do so while the cluster is down.

What Is Mixed Mode?

If you configure mixed mode (*HOSTNAME_ADDRESS_FAMILY* set to `ANY`), then the addresses used by the cluster, including the heartbeat, and Quorum Server addresses if any, can be IPv4 or IPv6 addresses. Serviceguard will first try to resolve a node's hostname to an IPv4 address, then, if that fails, will try IPv6.

Rules and Restrictions for Mixed Mode

IMPORTANT: See the latest version of the Serviceguard release notes for the most current information on these and other restrictions.

- The hostname resolution file on each node (for example, `/etc/hosts`) must contain entries for all the IPv4 and IPv6 addresses used throughout the cluster, including all *STATIONARY_IP* and *HEARTBEAT_IP* addresses as well any private addresses. There must be at least one IPv4 address in this file (in the case of `/etc/hosts`, the IPv4 loopback address cannot be removed).

  In addition, the file must contain the following entry:

  ```
  ::1 localhost ipv6-localhost ipv6-loopback
  ```

  For more information and recommendations about hostname resolution, see "Configuring Name Resolution" (page 159).

- You must use `$SGCONF/cmclnodelist`, not `~/.rhosts` or `/etc/hosts.equiv`, to provide root access to an unconfigured node.

  See "Allowing Root Access to an Unconfigured Node" (page 157) for more information.

- Hostname aliases are not supported for IPv6 addresses, because of operating system limitations.

  NOTE: This applies to all IPv6 addresses, whether *HOSTNAME_ADDRESS_FAMILY* is set to `IPV6` or `ANY`.

- Cross-subnet configurations are not supported.

  This also applies if *HOSTNAME_ADDRESS_FAMILY* is set to `IPV6`. See "Cross-Subnet Configurations" (page 29) for more information.

- VxVM, VxFS, and CVM/CFS are supported on HP Serviceguard A.11.20 April 2011 patches with Veritas Storage Foundation™ 5.1 SP1 and later.

NOTE: This also applies if *HOSTNAME_ADDRESS_FAMILY* is set to IPV6.

- HPVM is not supported.

  You cannot have a virtual machine that is either a node or a package if *HOSTNAME_ADDRESS_FAMILY* is set to ANY or IPV6.

Cluster Configuration Parameters

You need to define a set of cluster parameters. These are stored in the binary cluster configuration file, which is distributed to each node in the cluster. You configure these parameters by editing the cluster configuration template file created by means of the cmquerycl (1m) command, as described under "Configuring the Cluster " (page 177).

NOTE: See "Reconfiguring a Cluster" (page 278) for a summary of changes you can make while the cluster is running.

The following parameters must be configured:

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 (*).

NOTE: In addition, the following characters must not be used in the cluster name if you are using the Quorum Server: at-sign (@), equal-sign (=), or-sign (|), semicolon (;).

These characters are deprecated, meaning that you should not use them, even if you are not using the Quorum Server, because they will be illegal in a future Serviceguard release.

All other characters are legal. The cluster name can contain up to 39 characters (bytes).

△ **CAUTION:** Make sure that the cluster name is unique within the subnets configured on the cluster nodes; under some circumstances Serviceguard may not be able to detect a duplicate name and unexpected problems may result.

In particular make sure that two clusters with the same name do not use the same Quorum Server; this could result in one of the clusters failing to obtain the Quorum Server's arbitration services when it needs them, and thus failing to re-form.

HOSTNAME_ADDRESS_FAMILY

Specifies the Internet Protocol address family to which Serviceguard will try to resolve cluster node names and Quorum Server host names. Valid values are IPV4, IPV6, and ANY. The default is IPV4.

- IPV4 means Serviceguard will try to resolve the names to IPv4 addresses only.
- IPV6 means Serviceguard will try to resolve the names to IPv6 addresses only.
- ANY means Serviceguard will try to resolve the names to both IPv4 and IPv6 addresses.

IMPORTANT: See "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102) for important information. See also the latest Serviceguard release notes at http://www.hp.com/go/hpux-serviceguard-docs.

FIRST_CLUSTER_LOCK_VG, *SECOND_CLUSTER_LOCK_VG*

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

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

NOTE: Lock volume groups must also be defined in *VOLUME_GROUP* parameters in the cluster configuration file.

Can be changed while the cluster is running; see "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

QS_HOST

The fully-qualified hostname or IP address of a system outside the current cluster that is providing quorum service to the cluster. Can be (or resolve to) either an IPv4 or an IPv6 address if *HOSTNAME_ADDRESS_FAMILY* is set to ANY, but otherwise must match the setting of *HOSTNAME_ADDRESS_FAMILY*. This parameter is used only when you employ a Quorum Server for tie-breaking services in the cluster. You can also specify an alternate address (*QS_ADDR*) by which the cluster nodes can reach the Quorum Server.

For more information, see "Using a Quorum Server" (page 95) and "Specifying a Quorum Server" (page 180). See also "Configuring Serviceguard to Use the Quorum Server" in the latest version *HP Serviceguard Quorum Server Version A.04.00 Release Notes*, at http://www.hp.com/go/hpux-serviceguard-docs under HP Serviceguard Quorum Server Software.

IMPORTANT: See also"About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102) for important information about requirements and restrictions in an IPv6–only cluster.

Can be changed while the cluster is running; see "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

QS_ADDR

An alternate fully-qualified hostname or IP address for the Quorum Server. Can be (or resolve to) either an IPv4 or an IPv6 address if *HOSTNAME_ADDRESS_FAMILY* is set to ANY, but otherwise must match the setting of *HOSTNAME_ADDRESS_FAMILY*. This parameter is used

only if you use a Quorum Server and want to specify an address on an alternate subnet by which it can be reached. The alternate subnet need not use the same address family as *QS_HOST* if *HOSTNAME_ADDRESS_FAMILY* is set to ANY. For more information, see "Using a Quorum Server" (page 95) and "Specifying a Quorum Server" (page 180).

IMPORTANT: For special instructions that may apply to your version of Serviceguard and the Quorum Server see "Configuring Serviceguard to Use the Quorum Server" in the latest version *HP Serviceguard Quorum Server Version A.04.00 Release Notes*, at http://www.hp.com/go/hpux-serviceguard-docs under HP Serviceguard Quorum Server Software.

See also "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102) for important information about requirements and restrictions in an IPv6–only cluster.

Can be changed while the cluster is running; see "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

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). Minimum is 10,000,000 (10 seconds). Maximum is 2,147,483,647 (approximately 35 minutes).

Can be changed while the cluster is running; see "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

QS_TIMEOUT_EXTENSION

Optional parameter used to increase the time (in microseconds) to wait for a quorum server response.

The default quorum server timeout is calculated from the Serviceguard cluster parameter *MEMBER_TIMEOUT*. For clusters of two nodes, it is 0.1 * *MEMBER_TIMEOUT*; for more than two nodes it is 0.2 * *MEMBER_TIMEOUT*.

You can use the *QS_TIMEOUT_EXTENSION* to increase the time interval after which the current connection (or attempt to connect) to the Quorum Server is deemed to have failed; but do not do so until you have read the *HP Serviceguard Quorum Server Version A.04.00 Release Notes*, and in particular the following sections in that document: "About the QS Polling Interval and Timeout Extension", "Network Recommendations", and "Setting Quorum Server Parameters in the Cluster Configuration File".

Can be changed while the cluster is running; see "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

SITE_NAME

The name of a site to which nodes (see *NODE_NAME*) belong. Can be used only in a site-aware disaster-tolerant cluster, which requires Metrocluster (additional HP software); see

the documents listed under "Cross-Subnet Configurations" (page 29) for more information.

You can define multiple *SITE_NAME*s. *SITE_NAME* entries must precede any *NODE_NAME* entries. See also *SITE*.

NODE_NAME

The hostname of each system that will be a node in the cluster.

CAUTION: Make sure that the node name is unique within the subnets configured on the cluster nodes; under some circumstances Serviceguard may not be able to detect a duplicate name and unexpected problems may result.

Do not use the full domain name. For example, enter `ftsys9`, not `ftsys9.cup.hp.com`. A Serviceguard cluster can contain up to 16 nodes (though not in all third-party configurations; see "Veritas Cluster Volume Manager (CVM)" (page 82), and the latest Release Notes for your version of Serviceguard).

IMPORTANT: Node names must be 39 characters (bytes) or less, and are case-sensitive; for each node, the *NODE_NAME* in the cluster configuration file must exactly match the corresponding *node_name* in the package configuration file (see Chapter 6: "Configuring Packages and Their Services " (page 216)) and these in turn must exactly match the *hostname* portion of the name specified in the node's networking configuration. (Using the above example, `ftsys9` must appear in exactly that form in the cluster configuration and package configuration files, and as `ftsys9.cup.hp.com` in the DNS database).

The parameters immediately following *NODE_NAME* in this list (*SITE*, *NETWORK_INTERFACE*, *HEARTBEAT_IP*, *STATIONARY_IP*, *CLUSTER_LOCK_LUN*, *FIRST_CLUSTER_LOCK_PV*, *SECOND_CLUSTER_LOCK_PV*, *CAPACITY_NAME*, and *CAPACITY_VALUE*) apply specifically to the node identified by the preceding *NODE_NAME* entry.

SITE

The name of a site (defined by *SITE_NAME*) to which the node identified by the preceding *NODE_NAME* entry belongs. Can be used only in a site-aware disaster-tolerant cluster, which requires Metrocluster (additional HP software); see the documents listed under "Cross-Subnet Configurations" (page 29) for more information.

If *SITE* is used, it must be used for each node in the cluster (that is, all the nodes must be associated with some defined site, though not necessarily the same one).

If you are using *SITE*s, you can restrict the output of `cmviewcl (1m)` to a given site by means of the `-S <sitename>` option. In addition, you can configure a `site_preferred` or `site_preferred_manual` *failover_policy* (page 226) for a package.

NETWORK_INTERFACE

The name of each LAN that will be used for heartbeats or for user data on the node identified by the preceding *NODE_NAME*. An example is `lan0`. See also *HEARTBEAT_IP*, *STATIONARY_IP*, and "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102).

NOTE: Any subnet that is configured in this cluster configuration file as a *SUBNET* for IP monitoring purposes, or as a *monitored_subnet* in a package configuration file (or *SUBNET* in a legacy package; see "Package Configuration Planning " (page 120)) must be specified in the cluster configuration file via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. Similarly, any subnet that is used by a package for relocatable addresses should be configured into the cluster via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. For more information about relocatable addresses, see "Stationary and Relocatable IP Addresses " (page 64) and the descriptions of the package *ip_* parameters (page 231).

IMPORTANT: See "Cross-Subnet Configurations" (page 29) for important information about requirements for such configurations.

For information about changing the configuration online, see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

HEARTBEAT_IP

IP notation indicating this node's connection to a subnet that will carry the cluster heartbeat.

NOTE: Any subnet that is configured in this cluster configuration file as a *SUBNET* for IP monitoring purposes, or as a *monitored_subnet* in a package configuration file (or *SUBNET* in a legacy package; see "Package Configuration Planning " (page 120)) must be specified in the cluster configuration file via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. Similarly, any subnet that is used by a package for relocatable addresses should be configured into the cluster via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. For more information about relocatable addresses, see "Stationary and Relocatable IP Addresses " (page 64) and the descriptions of the package *ip_* parameters (page 231).

If *HOSTNAME_ADDRESS_FAMILY* is set to `IPV4` or `ANY`, a heartbeat IP address can be either an IPv4 or an IPv6 address, with the exceptions noted below. If *HOSTNAME_ADDRESS_FAMILY* is set to `IPV6`, all heartbeat IP addresses must be IPv6 addresses.

For more information about the IPv6 address format, see "IPv6 Address Types" (page 364). Heartbeat IP addresses

on a given subnet must all be of the same type: IPv4 or IPv6 site-local or IPv6 global.

For information about changing the configuration online, see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

Heartbeat configuration requirements:

A minimum Serviceguard configuration on HP-UX 11i v2 or 11i v3 needs two network interface cards for the heartbeat in all cases, using one of the following configurations:

- Two heartbeat subnets; *or*
- One heartbeat subnet with a standby; *or*
- One heartbeat subnet using APA with two physical ports in hot standby mode or LAN monitor mode.

You cannot configure more than one heartbeat IP address on an interface; only one *HEARTBEAT_IP* is allowed for each *NETWORK_INTERFACE*.

Do not mix APA LAN Monitor or hot standby modes on the same network interfaces that are configured for Serviceguard local LAN failover. Since these monitoring and failover functions work differently and do not communicate with each other, unexpected failover results can occur. HP supports mixing these LAN HA methods for different subsets of network interfaces within a cluster, but does not support mixing them for the same set of interfaces.

NOTE: The Serviceguard `cmapplyconf`, `cmcheckconf`, and `cmquerycl` commands check that these minimum requirements are met, and produce a warning if they are not met at the immediate network level. If you see this warning, you need to check that the requirements are met in your overall network configuration.

If you are using virtual machine guests as nodes, you have a valid configuration (and can ignore the warning) if there is one heartbeat network on the guest, backed by a network on the host using APA with two trunk members (HPVM), or using NIC bonding in high availability mode (or mode 1) with two slaves (VMware ESX Server).

For information about changing the configuration online, see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

Considerations for cross-subnet:

IP addresses for a given heartbeat path are usually on the same subnet on each node, but it is possible to configure the heartbeat on multiple subnets such that the heartbeat is carried on one subnet for one set of nodes and another subnet for others, with the subnets joined by a router.

This is called a cross-subnet configuration, and in this case *at least two heartbeat paths* must be configured for each cluster node. In addition, each heartbeat subnet on each node must be physically routed separately to the heartbeat

subnet on another node (that is, each heartbeat path must be physically separate). See "Cross-Subnet Configurations" (page 29) for more information.

NOTE: *Limitations:*

- Because Veritas Cluster File System from Symantec (CFS) requires link-level traffic communication (LLT) among the nodes, Serviceguard cannot be configured in cross-subnet configurations with CFS alone. But CFS *is* supported in specific cross-subnet configurations with Serviceguard and HP add-on products; see the documentation listed under "Cross-Subnet Configurations" (page 29) for more information.
- IPv6 heartbeat subnets are not supported in a cross-subnet configuration.

Considerations for CVM:

- For Veritas CVM 4.1 or later, multiple heartbeats are permitted, and you must configure either multiple heartbeat subnets or a single heartbeat subnet with a standby. HP recommends multiple heartbeats.
- You cannot change the heartbeat configuration while a cluster that uses CVM is running.
- You can use an IPv6 heartbeat subnet with Veritas CVM/CFS on HP Serviceguard A.11.20 April 2011 patches with Veritas Storage Foundation™ 5.1 SP1 and later.

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 timeout 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`

This node's IP address on each subnet that does not carry the cluster heartbeat, but is monitored for packages.

NOTE: Any subnet that is configured in this cluster configuration file as a *SUBNET* for IP monitoring purposes, or as a *monitored_subnet* in a package configuration file (or *SUBNET* in a legacy package; see "Package Configuration Planning " (page 120)) must be specified in the cluster configuration file via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. Similarly, any subnet that is used by a package for relocatable addresses should be configured into the cluster via *NETWORK_INTERFACE* and either *STATIONARY_IP* or *HEARTBEAT_IP*. For more information about relocatable addresses, see "Stationary and Relocatable IP Addresses " (page 64) and the descriptions of the package *ip_* parameters (page 231).

If you want to separate application data from heartbeat messages, define one or more monitored non-heartbeat subnets here. You can identify any number of subnets to be monitored.

IMPORTANT: In a cross-subnet configuration, each package subnet configured on an interface (NIC) must have a standby interface connected to the local bridged network. See "Cross-Subnet Configurations" (page 29) for important information about requirements for such configurations.

If *HOSTNAME_ADDRESS_FAMILY* is set to `IPV4` or `ANY`, a stationary IP address can be either an IPv4 or an IPv6 address. If *HOSTNAME_ADDRESS_FAMILY* is set to `IPV6`, all the IP addresses used by the cluster must be IPv6 addresses. For information about the IPv6 address format, see "IPv6 Address Types" (page 364).

For information about changing the configuration online, see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

CLUSTER_LOCK_LUN

The path on this node for the LUN used for the cluster lock. Used only if a lock LUN is used for tie-breaking services.

Enter the path as it appears on each node in the cluster (the same physical device may have a different name on each node). The path must identify a block device.

You cannot create a dual cluster-lock configuration using LUNs.

See "Setting Up a Lock LUN" (page 165) and "Specifying a Lock LUN" (page 180) for more information.

Can be changed while the cluster is running; see "Updating the Cluster Lock LUN Configuration Online" (page 282). See also "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

FIRST_CLUSTER_LOCK_PV, *SECOND_CLUSTER_LOCK_PV*

The path on this node for the physical volume within the cluster-lock Volume Group that will have the cluster lock written on it (see the entry for *FIRST_CLUSTER_LOCK_VG* and *SECOND_CLUSTER_LOCK_VG* near the beginning of

this list). Used only if a lock disk is used for tie-breaking services. Use *FIRST_CLUSTER_LOCK_PV* for the first physical lock volume and *SECOND_CLUSTER_LOCK_PV* for the second physical lock volume, if any. If there is a second physical lock volume, *SECOND_CLUSTER_LOCK_PV* must be on a separate line. These parameters are used only when you employ a lock disk for tie-breaking services in the cluster.

If you need to change the default entries, make sure you enter the physical volume name as it appears on each node in the cluster (the same physical volume may have a different path name on each node).

NOTE: If you used the `-L lock_vg:lock_pv` option of `cmquerycl (1m)` to specify the cluster lock disk, the correct node-specific paths will already be in the file.

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 (bytes).

For information about changing the configuration while the cluster is running, see "Updating the Cluster Lock Disk Configuration Online" (page 281). See also "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

CAPACITY_NAME, *CAPACITY_VALUE*

Node capacity parameters. Use the *CAPACITY_NAME* and *CAPACITY_VALUE* parameters to define a capacity for this node. Node capacities correspond to package weights; node capacity is checked against the corresponding package weight to determine if the package can run on that node.

CAPACITY_NAME name can be any string that starts and ends with an alphanumeric character, and otherwise contains only alphanumeric characters, dot (.), dash (-), or underscore (_). Maximum length is 39 characters. *CAPACITY_NAME* must be unique in the cluster.

CAPACITY_VALUE specifies a value for the *CAPACITY_NAME* that precedes it. It must be a floating-point value between 0 and 1000000. Capacity values are arbitrary as far as Serviceguard is concerned; they have meaning only in relation to the corresponding package weights.

Capacity definition is optional, but if *CAPACITY_NAME* is specified, *CAPACITY_VALUE* must also be specified; *CAPACITY_NAME* must come first.

NOTE: `cmapplyconf` will fail if any node defines a capacity and any package has `min_package_node` as its *failover_policy* (page 226) or `automatic` as its *failback_policy* (page 227).

To specify more than one capacity for a node, repeat these parameters for each capacity. You can specify a maximum

of four capacities per cluster, unless you use the reserved *CAPACITY_NAME* `package_limit`; in that case, you can use only that capacity throughout the cluster.

For all capacities other than `package_limit`, the default weight for all packages is zero, though you can specify a different default weight for any capacity other than `package_limit`; see the entry for *WEIGHT_NAME* and *WEIGHT_DEFAULT* later in this list.

See "About Package Weights" (page 135) for more information.

Can be changed while the cluster is running; will trigger a warning if the change would cause a running package to fail.

MEMBER_TIMEOUT

The amount of time, in microseconds, after which Serviceguard declares that the node has failed and begins re-forming the cluster without this node.

Default value: 14 seconds (14,000,000 microseconds).

This value leads to a failover time of between approximately 18 and 22 seconds, if you are using a Quorum Server, or a Fiber Channel cluster lock, or no cluster lock. Increasing the value to 25 seconds increases the failover time to between approximately 29 and 39 seconds. The time will increase by between 5 and 13 seconds if you are you using a SCSI cluster lock or dual Fibre Channel cluster lock).

Maximum supported value: 300 seconds (300,000,000 microseconds).

If you enter a value greater than 60 seconds (60,000,000 microseconds), `cmcheckconf` and `cmapplyconf` will note the fact, as confirmation that you intend to use a large value.

Minimum supported values:

- 3 seconds for a cluster with more than one heartbeat subnet.
- 8 seconds for a cluster with more than one heartbeat using CFS (see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190) for more information about CFS; see also the "Considerations for CVM" under *HEARTBEAT_IP* earlier in this section).
- 14 seconds for a cluster that has only one Ethernet heartbeat LAN
- 22 seconds for a cluster that has only one IP over Infiniband (IPoIP) heartbeat LAN

NOTE:

- For most clusters that use an LVM cluster lock or lock LUN, a minimum *MEMBER_TIMEOUT* of 14 seconds is appropriate.
- For most clusters that use a *MEMBER_TIMEOUT* value lower than 14 seconds, a quorum server is more appropriate than a lock disk or lock LUN.

  The cluster will fail if the time it takes to acquire the disk lock exceeds 0.2 times the *MEMBER_TIMEOUT*. This means that if you use a disk-based quorum device (lock disk or lock LUN), you must be certain that the nodes in the cluster, the connection to the disk, and the disk itself can respond quickly enough to perform 10 disk writes within 0.2 times the *MEMBER_TIMEOUT*.

With the lowest supported value of 3 seconds, a failover time of 4 to 5 seconds can be achieved.

NOTE: The failover estimates provided here apply to the Serviceguard component of failover; that is, the package is expected to be up and running on the adoptive node in this time, but the application that the package runs may take more time to start.

Keep the following guidelines in mind when deciding how to set the value.

Guidelines: You need to decide whether it's more important for your installation to have *fewer* (but slower) cluster re-formations, or *faster* (but possibly more frequent) re-formations:

- To ensure the fastest cluster re-formations, use the minimum value applicable to your cluster. But keep in mind that this setting will lead to a cluster re-formation, and to the node being removed from the cluster and rebooted, if a system hang or network load spike prevents the node from sending a heartbeat signal within the *MEMBER_TIMEOUT* value. More than one node could be affected if, for example, a network event such as a broadcast storm caused kernel interrupts to be turned off on some or all nodes while the packets are being processed, preventing the nodes from sending and processing heartbeat messages.

  See "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320) for troubleshooting information.
- For fewer re-formations, use a setting in the range of 10 to 25 seconds (10,000,000 to 25,000,000 microseconds), keeping in mind that a value larger than the default will lead to slower re-formations than the default. A value in this range is appropriate for most installations

See also "What Happens when a Node Times Out" (page 85), "Cluster Daemon: cmcld" (page 39), and the white paper *Optimizing Failover Time in a Serviceguard Environment (version A.11.19 and later)* on http://www.hp.com/go/hpux-serviceguard-docs.

Can be changed while the cluster is running.

AUTO_START_TIMEOUT

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

Default is 600,000,000 microseconds.

Can be changed while the cluster is running.

NETWORK_POLLING_INTERVAL

Specifies how frequently the networks configured for Serviceguard are checked.

Default is 2,000,000 microseconds (2 seconds). This means that the network manager will poll each network interface every 2 seconds, to make sure it can still send and receive information.

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

IMPORTANT: HP strongly recommends using the default. Changing this value can affect how quickly the link-level and IP-level monitors detect a network failure. See "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66), "Monitoring LAN Interfaces and Detecting Failure: IP Level" (page 70), and "Reporting Link-Level and IP-Level Failures" (page 73).

Can be changed while the cluster is running.

CONFIGURED_IO_TIMEOUT_EXTENSION

The number of microseconds by which to increase the time Serviceguard waits after detecting a node failure, so as to ensure that all pending I/O on the failed node has ceased.

This parameter must be set in the following cases.

- For extended-distance clusters using software mirroring across data centers over links between iFCP switches; it must be set to the switches' maximum *R_A_TOV* value.

  NOTE: *CONFIGURED_IO_TIMEOUT_EXTENSION* is supported only with iFCP switches that allow you to get their *R_A_TOV* value.

- For switches and routers connecting an NFS server and cluster-node clients that can run packages using the NFS-mounted file system; see "Planning for NFS-mounted File Systems" (page 125).

  To set the value for the *CONFIGURED_IO_TIMEOUT_EXTENSION*, you must

first determine the Maximum Bridge Transit Delay (MBTD) for each switch and router. The value should be in the vendors' documentation. Set the *CONFIGURED_IO_TIMEOUT_EXTENSION* to the *sum of the values* for the switches and routers. If there is more than one possible path between the NFS server and the cluster nodes, sum the values for each path and use the largest number.

△ **CAUTION:** Serviceguard supports NFS-mounted file systems only over switches and routers that support MBTD. If you are using NFS-mounted file systems, you must set *CONFIGURED_IO_TIMEOUT_EXTENSION* as described here.

For a fuller discussion of MBTD, see the white paper *Support for NFS as a file system type with Serviceguard 11.20 on HP-UX 11i v3* which you can find at http://www.hp.com/go/hpux-serviceguard-docs.

- For clusters in which both of the above conditions apply.

  In this case, set the *CONFIGURED_IO_TIMEOUT_EXTENSION* to the higher of the two values you get from following the instructions in the preceding two bullets.

Default is 0.

NETWORK_FAILURE_DETECTION

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

- `INOUT`
- `INONLY_OR_INOUT`

The default is `INOUT`.

See "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66) for more information.

Can be changed while the cluster is running.

NETWORK_AUTO_FAILBACK

How Serviceguard will handle the recovery of the primary LAN interface after it has failed over to the standby interface because of a link level failure. Valid values are `YES` and `NO`. `YES` means the IP address(es) will fail back to the primary LAN interface from the standby when the primary LAN interface recovers at the link level. `NO` means the IP address(es) will fail back to the primary LAN interface only when you use `cmmodnet (1m)` to re-enable the interface.

Default is `YES`.

SUBNET

IP address of a cluster subnet for which IP Monitoring can be turned on or off (see *IP_MONITOR*). The subnet must be configured into the cluster, via *NETWORK_INTERFACE* and either *HEARTBEAT_IP* or *STATIONARY_IP*. All entries for *IP_MONITOR* and *POLLING_TARGET* apply to this subnet until the next *SUBNET* entry; *SUBNET* must be the first of each trio.

By default, each of the cluster subnets is listed under *SUBNET*, and, if at least one gateway is detected for that subnet, *IP_MONITOR* is set to ON and *POLLING_TARGET* entries are populated with the gateway addresses, enabling target polling; otherwise the subnet is listed with *IP_MONITOR* set to OFF.

See "Monitoring LAN Interfaces and Detecting Failure: IP Level" (page 70) for more information.

Can be changed while the cluster is running; must be removed, with its accompanying *IP_MONITOR* and *POLLING_TARGET* entries, if the subnet in question is removed from the cluster configuration.

IP_MONITOR

Specifies whether or not the subnet specified in the preceding *SUBNET* entry will be monitored at the IP layer.

To enable IP monitoring for the subnet, set *IP_MONITOR* to ON; to disable it, set it to OFF.

By default *IP_MONITOR* is set to ON if a gateway is detected for the *SUBNET* in question, and *POLLING_TARGET* entries are populated with the gateway addresses, enabling target polling. See the *POLLING_TARGET* description that follows for more information.

HP recommends you use target polling because it enables monitoring beyond the first level of switches, but if you want to use peer polling instead, set *IP_MONITOR* to ON for this *SUBNET*, but do not use *POLLING_TARGET* (comment out or delete any *POLLING_TARGET* entries that are already there).

If a network interface in this subnet fails at the IP level and *IP_MONITOR* is set toON, the interface will be marked down. If it is set to OFF, failures that occur only at the IP-level will not be detected.

Can be changed while the cluster is running; must be removed if the preceding *SUBNET* entry is removed.

POLLING_TARGET

The IP address to which polling messages will be sent from all network interfaces on the subnet specified in the preceding *SUBNET* entry, if *IP_MONITOR* is set to ON. This is called target polling.

Each subnet can have multiple polling targets; repeat *POLLING_TARGET* entries as needed.

If *IP_MONITOR* is set to ON, but no *POLLING_TARGET* is specified, polling messages are sent between network interfaces on the same subnet (peer polling). HP recommends you use target polling; see "How the IP Monitor Works" (page 71) for more information.

NOTE: `cmquerycl (1m)` detects first-level routers in the cluster (by looking for gateways in each node's routing table) and lists them here as polling targets. If you run `cmquerycl` with the `-w full` option (for full network probing) it will also verify that the gateways will work correctly for monitoring purposes.

Can be changed while the cluster is running; must be removed if the preceding *SUBNET* entry is removed.

MAX_CONFIGURED_PACKAGES

This parameter sets the maximum number of packages that can be configured in the cluster.

The minimum value is 0, and the maximum value is 300. The default value is 300, and you can change it without halting the cluster.

WEIGHT_NAME, *WEIGHT_DEFAULT*

Default value for this weight for all packages that can have weight; see "Rules and Guidelines" (page 141) under"About Package Weights" (page 135). *WEIGHT_NAME* specifies a name for a weight that *exactly corresponds* to a *CAPACITY_NAME* specified earlier in the cluster configuration file. (A package has weight; a node has capacity.) The rules for forming *WEIGHT_NAME* are the same as those spelled out for *CAPACITY_NAME* earlier in this list.

These parameters are optional, but if they are defined, *WEIGHT_DEFAULT* must follow *WEIGHT_NAME*, and must be set to a floating-point value between 0 and 1000000. If they are not specified for a given weight, Serviceguard will assume a default value of zero for that weight. In either case, the default can be overridden for an individual package via the *weight_name* and *weight_value* parameters in the package configuration file.

For more information and examples, see "Defining Weights" (page 139).

IMPORTANT: *CAPACITY_NAME*, *WEIGHT_NAME*, and *weight_value* must all match exactly.

NOTE: A weight (*WEIGHT_NAME*, *WEIGHT_DEFAULT*) has no meaning on a node unless a corresponding capacity (*CAPACITY_NAME*, *CAPACITY_VALUE*) is defined for that node.

For the reserved weight and capacity `package_limit`, the default weight is always one. This default cannot be changed in the cluster configuration file, but it can be overridden for an individual package in the package configuration file.

`cmapplyconf` will fail if you define a default for a weight but do not specify a capacity of the same name for at least one node in the cluster. You can define a maximum of four *WEIGHT_DEFAULT*s per cluster.

Can be changed while the cluster is running.

Access Control Policies (also known as Role Based Access)	For each policy, specify `USER_NAME`, `USER_HOST`, and `USER_ROLE`. Policies set in the configuration file of a cluster and its packages must not be conflicting or redundant. For more information, see "Setting up Access-Control Policies" (page 185).
`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. The volume group name can have up to 39 characters (bytes).

Cluster Configuration: Next Step

When you are ready to configure the cluster, proceed to "Configuring the Cluster " (page 177). If you find it useful to record your configuration ahead of time, use the worksheet in Appendix E.

Package Configuration Planning

Planning for packages involves assembling information about each group of highly available services.

NOTE: As of Serviceguard A.11.18, there is a new and simpler way to configure packages. This method allows you to build packages out of smaller modules, and eliminates the separate package control script and the need to distribute it manually; see Chapter 6: "Configuring Packages and Their Services " (page 216) for complete instructions.

This manual refers to packages produced by the newer method as modular packages, and to packages produced by the older method as legacy packages.

The discussion that follows assumes you will be using the modular method. For information and instructions on creating and maintaining older packages, see "Configuring a Legacy Package" (page 289).

The document *Framework for HP Serviceguard Toolkits* provides a guide to integrating an application with Serviceguard, and includes a suite of customizable scripts intended for use with legacy packages. This document is included in the Serviceguard Developer's Toolbox, which you can download free of charge from http://www.hp.com/go/softwaredepot.

NOTE: As of the date of this manual, the *Framework for HP Serviceguard Toolkits* deals specifically with legacy packages.

Logical Volume and File System Planning

NOTE: LVM Volume groups that are to be activated by packages must also be defined as cluster-aware in the cluster configuration file. See "Cluster Configuration Planning " (page 99). Disk groups (for Veritas volume managers) that are to be activated by packages must be defined in the package configuration file, described below.

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 for this to happen, you must 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 file systems at different times.

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

Create an entry for each logical volume, indicating its use for a file system or for a raw device. Don't forget to comment out the lines (using the # character as shown).

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

Planning Veritas Cluster Volume Manager (CVM) and Cluster File System (CFS)

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information on support for CVM and CFS: http://www.hp.com/go/hpux-serviceguard-docs.

For a failover package that uses CVM or CFS, you configure system multi-node packages to handle the volume groups and file systems.

CAUTION: Serviceguard manages Veritas processes, specifically `gab` and `LLT`, through system multi-node packages. As a result, the Veritas administration commands such as `gabconfig`, `llthosts`, and `lltconfig` should only be used in display mode, for example `gabconfig -a`. You could crash nodes or the entire cluster if you use Veritas commands such as the `gab*` or `llt*` commands to configure these components or affect their runtime behavior.

CVM 4.1 and later *without* CFS

Veritas Cluster Volume Manager 4.1 and later uses the system multi-node package `SG-CFS-pkg` to manage the cluster's volumes.

CVM 4.1 and later requires you to configure multiple heartbeat networks, or a single heartbeat with a standby. The following interfaces are supported as heartbeat networks — APA (CVM 5.0.1 and later) and VLAN (CVM 4.1 and later).

CVM 4.1 and later *with* CFS

CFS (Veritas Cluster File System) is supported for use with Veritas Cluster Volume Manager Version 4.1 and later.

The system multi-node package `SG-CFS-pkg` manages the cluster's volumes. When you use CFS, Serviceguard enables you to create and manage multi-node packages for disk groups and mount points using modular style packages or legacy style packages.

To create modular CVM disk group and CFS mount point packages, use the `cmmakepkg` command and edit the parameters in the package configuration file. You can choose to name the packages. You can also use the Serviceguard Manager to create the modular CFS packages. See the Serviceguard Manager online help for more information. Alternatively, you can create CFS legacy disk group and mount point packages with the `cfs` family of commands; do *not* edit the configuration file. Serviceguard names the packages `SG-CFS-DG-id#` and `SG-CFS-MP-id#`, respectively, and automatically increments their ID numbers.

NOTE: It is highly recommend to use the modular style of packaging as they offer flexibility and can be managed better compared to the legacy style of packaging. For differences between the two, see "Modular CFS packages v/s Legacy CFS packages" (page 190). For more information on modular CFS packages, see "Managing Disk Groups and Mount Points Using Modular Packages" (page 195).

CVM 4.1 and later requires you to configure multiple heartbeat networks, or a single heartbeat with a standby. The following interfaces are supported as heartbeat networks — APA (CVM 5.0.1 and later) and VLAN (CVM 4.1 and later).

You create a chain of package dependencies for application failover packages and the non-failover packages:

1. The failover package's applications should not run on a node unless the mount point packages are already running.

   In the package's configuration file, you fill out the dependency parameter to specify the requirement that the disk group is `UP` on the `SAME_NODE`.

2. The disk group packages should not run unless the CFS system multi-node package (`SG-CFS-pkg`) is running to manage the volumes.

3. The mount point packages should not run unless the disk group packages are running.

   If the disk groups and mount points are in separate packages, specify the dependencies on the disk group packages in the configuration file.

CAUTION: Once you create the modular CVM disk group and CFS mount point packages, you must administer the cluster with `cmcheckconf`, `cmapplyconf`, `cmrunpkg`, `cmmodpkg`, and `cmrunpkg` commands. If you create the legacy CFS packages, you must administer the cluster with CFS commands, including `cfsdgadm`, `cfsmntadm`, `cfsmount`, and `cfsumount`.

You must not use the HP-UX `mount` or `umount` command to provide or remove access to a shared file system in a CFS environment; using these HP-UX commands under these circumstances is not supported. Use `cfsmount` and `cfsumount` instead. If you use the HP-UX `mount` and `umount` commands, serious problems could occur, such as writing to the local file system instead of the cluster file system. Non-CFS commands could cause conflicts with subsequent CFS command operations on the file system or the Serviceguard packages, and will not create an appropriate multi-node package, which means cluster packages will not be aware of file system changes.

NOTE: The Disk Group (DG) and Mount Point (MP) multi-node packages do not monitor the *health of* the disk group and mount point. They check that the application packages that depend on them have *access to* the disk groups and mount points. If the dependent application package loses access and cannot read and write to the disk, it will fail, but that will not cause the DG or MP multi-node package to fail.

4. You create the CFS package, `SG-CFS-pkg`, with the `cfscluster` command. It is a system multi-node package that regulates the volumes used by CVM 4.1 and later. System multi-node packages cannot be dependent on any other package.

About the Volume Monitor

Simply monitoring each physical disk in a Serviceguard cluster does not provide adequate monitoring for volumes managed by Veritas Volume Manager from Symantec (VxVM) or logical volumes managed by HP-UX Logical Volume Manager (LVM), because a physical volume failure is not always a critical failure that triggers failover (for example, the failure of a single physical disk within a mirrored volume is not considered critical). For this reason, it can be very difficult to determine which physical disks must be monitored to ensure that a storage volume is functioning properly. The HP Serviceguard Volume Monitor provides a means for simple and effective monitoring of storage volumes.

IMPORTANT: Check the latest version of the release notes (at the address given in the preface to this manual) for information about Serviceguard support for the volume monitor.

Using the Volume Monitor

NOTE: For LVM, using this monitor is an alternative to using Event Monitoring Service (EMS) resource dependencies; see "Using EMS to Monitor Volume Groups" (page 96). EMS does not currently provide a monitor for VxVM.

Configure the Volume Monitor as a service in a package that requires access to a VxVM or LVM storage volume.

The package can be a failover package or multi-node package. For example, you can configure the monitor as a service in a failover package to monitor a storage volume (or multiple storage volumes) required by the package application. Alternatively, the monitor could be used in a multi-node package to monitor identically-named root, boot, or swap volumes on cluster nodes. Because the root, boot, and swap volumes are critical to the functioning of the node , the service should be configured with *`service_fail_fast_enabled`* (page 233) set to `yes`.

When a monitored volume fails or becomes inaccessible, the monitor service will exit, causing the package to fail on the current node. The package's failure behavior depends on its configured settings. For prompt recovery, HP recommends setting the value of *service_restart* (page 233) for the monitoring service to `none`.

To ensure that a package requiring a storage volume does not attempt to start on or fail over to a node where the storage volume is unavailable, the monitor service may be configured in a separate package, and a package dependency may be used to ensure that the required package is running, indicating the storage is available. Depending on the configuration, the monitor package could be a multi-node or failover package, and would be required to be running by the storage volume-dependent application package. Alternatively, if you are using EMS, please be aware that EMS resource dependencies operate in this fashion – a package will not be run unless the specified EMS resources conditions are met.

NOTE: When using the volume monitor to monitor LVM logical volumes, you need to make sure that the logical volume timeout value is properly configured. This value should be configured to be at least one second less than the *poll-interval* specified in the monitor service command. I/O requests to logical volumes with no timeout set may block indefinitely. See "Setting Logical Volume Timeouts" (page 171) for more information.

Command Syntax

The syntax for the monitoring command, `cmvolmond`, is as follows:

```
cmvolmond [-h, --help] [-v, --version]
[-O, --log-file <log_file>
[-D, --log-level <1-7>
[-t, --poll-interval <seconds>
<volume_path> [<volume_path> ...]
```

A brief description of each parameter follows:

`-h` or `--help`

Displays the usage, as listed above, and exits.

NOTE: Do not include the help or version parameters in your service command; this will result in immediate package failure at runtime.

`-v or --version`

Displays the monitor version and exits.

`-O or --log-file`

Specifies a file for logging (log messages are printed to the console by default).

`-D or --log-level`

Specifies the log level. The level of detail logged is directly proportional to the numerical value of the log level. That is, a log level of 7 will provide the greatest amount of log information. The default log level is 0.

`-t or --poll-interval`

Specifies the interval between volume probes. You can specify a polling interval of as little as `1` (one second), but bear in mind that a short polling interval (less than 10 seconds) may impair system performance if you are monitoring a large number of volumes. HP recommends a polling interval of at least 10 seconds if 50 or more volumes are being monitored by a single service command. The default polling interval is 60 seconds. In the event of a failed read attempt on a storage volume, the monitor service will terminate after a single poll interval. If a read attempt never completes, the monitor service will terminate after six such consecutive read attempts (a duration of up to six poll intervals).

`volume_path`

The full path to at least one VxVM volume or LVM logical volume device file for monitoring (required). The pathname must identify a block device file.

Examples

`/usr/sbin/cmvolmond -O /log/monlog.log -D 3 /dev/vx/dsk/cvm_dg0/lvol2`

This command monitors a single VxVM volume, `/dev/vx/dsk/cvm_dg0/lvol2`, at log level 3, with a polling interval of 60 seconds, and prints all log messages to `/log/monlog.log`.

`/usr/sbin/cmvolmond /dev/vg01/lvol1 /dev/vg01/lvol2`

This command monitors two LVM logical volumes at the default log level of 0, with a polling interval of 60 seconds, and prints all log messages to the console.

`/usr/sbin/cmvolmond -t 10 /dev/vg00/lvol1`

This command monitors the LVM root logical volume at log level 0, with a polling interval of 10 seconds, and prints all log messages to the console (package log).

Scope of Monitoring

The Volume Monitor detects the following failures:

- Failure of the last link to a storage device or set of devices critical to volume operation
- Failure of a storage device or set of devices critical to volume operation
- An unexpected detachment, disablement, or deactivation of a volume

The Volume Monitor *does not* detect the following failures:

- Failure of a redundant link to a storage device or set of devices where a working link remains
- Failure of a mirror or mirrored plex within a volume (assuming at least one mirror or plex is functional)
- Corruption of data on a monitored volume.

Planning for NFS-mounted File Systems

As of Serviceguard A.11.20, you can use NFS-mounted (imported) file systems as shared storage in packages.

The same package can mount more than one NFS-imported file system, and can use both cluster-local shared storage and NFS imports.

The following rules and restrictions apply.

- NFS mounts are supported for modular, failover packages. It is now possible (as of A.11.20 April 2011 patch release) to create a Multi-Node Package that uses an NFS file share, and this is useful if you want to create a HP Integrity Virtual Machine (HPVM) in a Serviceguard Package, where the virtual machine itself uses a remote NFS share as backing store.

  For details on how to configure NFS as a backing store for HPVM, see the *HP Integrity Virtual Machines 4.3: Installation, Configuration, and Administration* guide at http://www.hp.com/go/virtualization-manuals `–> HP Integrity Virtual Machines and Online VM Migration.`

  See Chapter 6 (page 216) for a discussion of types of packages.

- So that Serviceguard can ensure that all I/O from a node on which a package has failed is flushed before the package restarts on an adoptive node, all the network switches and routers between the NFS server and client must support a worst-case timeout, after which packets and frames are dropped. This timeout is known as the Maximum Bridge Transit Delay (MBTD).

IMPORTANT: Find out the MBTD value for each affected router and switch from the vendors' documentation; determine all of the possible paths; find the worst case sum of the MBTD values on these paths; and use the resulting value to set the Serviceguard `CONFIGURED_IO_TIMEOUT_EXTENSION` parameter. For instructions, see the discussion of this parameter under "Cluster Configuration Parameters " (page 105).

Switches and routers that do not support MBTD value must not be used in a Serviceguard NFS configuration. This might lead to delayed packets that in turn could lead to data corruption.

- Networking among the Serviceguard nodes must be configured in such a way that a single failure in the network does not cause a package failure.
- Only NFS client-side locks (local locks) are supported.

  Server-side locks are not supported.
- Because exclusive activation is not available for NFS-imported file systems, you should take the following precautions to ensure that data is not accidentally overwritten.
  - The server should be configured so that only the cluster nodes have access to the file system.
  - The NFS file system used by a package must not be imported by any other system, including other nodes in the cluster. The only exception to this restriction is when you want to use the NFS file system as a backing store for HPVM. In this case, the NFS file system is configured as a multi-node package and is imported on more than one node in the cluster.
  - The nodes should not mount the file system on boot; it should be mounted only as part of the startup for the package that uses it.
  - The NFS file system should be used by only one package.
  - While the package is running, the file system should be used exclusively by the package.
  - If the package fails, do not attempt to restart it manually until you have verified that the file system has been unmounted properly.

In addition, you should observe the following guidelines.

- CacheFS and AutoFS should be disabled on all nodes configured to run a package that uses NFS mounts.

  For more information, see the *NFS Services Administrator's Guide HP-UX 11i version 3* at http://www.hp.com/go/hpux-networking-docs.
- HP recommends that you avoid a single point of failure by ensuring that the NFS server is highly available.

  NOTE: If network connectivity to the NFS Server is lost, the applications using the imported file system may hang and it may not be possible to kill them. If the package attempts to halt at this point, it may not halt successfully.
- Do not use the automounter; otherwise package startup may fail.
- If storage is directly connected to all the cluster nodes and shared, configure it as a local file system rather than using NFS.
- An NFS file system should not be mounted on more than one mount point at the same time.
- Access to an NFS file system used by a package should be restricted to the nodes that can run the package.

For more information, see the white paper *Using NFS as a file system type with Serviceguard 11.20 on HP-UX 11i v3* which you can find at http://www.hp.com/go/hpux-serviceguard-docs.

This paper includes instructions for setting up a sample package that uses an NFS-imported file system.

See also the description of *fs_name* (page 238), *fs_type* (page 238), and the other file system-related package parameters.

Planning for Expansion

You can add packages to a running cluster. This process is described in "Cluster and Package Maintenance" (page 248).

When adding packages, be sure not to exceed the value of *max_configured_packages* as defined in the cluster configuration file; see "Cluster Configuration Parameters " (page 105). You can modify this parameter while the cluster is running if you need to.

Choosing Switching and Failover Behavior

To determine the failover behavior of a failover package (see "Package Types" (page 48)), you define the policy that governs where Serviceguard will automatically start up a package that is not running. In addition, you define a failback policy that determines whether a package will be automatically returned to its primary node when that is possible.

The following table describes different types of failover behavior and the settings in the package configuration file that determine each behavior. See "Package Parameter Explanations" (page 222) for more information.

Table 7 Package Failover Behavior

Switching Behavior	Parameters in Configuration File
Package switches normally after detection of service, network, or EMS failure, or when a configured resource dependency is not met. Halt script runs before switch takes place. (Default)	• *node_fail_fast_enabled* set to no. (Default) • *service_fail_fast_enabled* set to NO for all services. (Default) • *auto_run* set to yes for the package. (Default)
Package fails over to the node with the fewest active packages.	• *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)
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.
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) • failover_policy set to configured_node. (Default)
All packages switch following a system reset (an immediate halt without a graceful shutdown) on the node when a specific service fails. Halt scripts are not run.	• *service_fail_fast_enabled* set to yes for a specific service. • *auto_run* set to yes for all packages.
All packages switch following a system reset on the node when any service fails. An attempt is first made to reboot the system prior to the system reset.	• *service_fail_fast_enabled* set to yes for *all* services. • *auto_run* set to yes for all packages.

Failover packages can be also configured so that IP addresses switch from a failed LAN card to a standby LAN card on the same node and the same physical subnet. To manage this behavior, use the parameter *local_lan_failover_allowed* (page 229) in the package configuration file. (yes, meaning enabled, is the default.)

Parameters for Configuring EMS Resources

NOTE: The default form for parameter names and literal values in the modular package configuration file is lower case; for legacy packages the default is upper case. There are no compatibility issues; Serviceguard is case-insensitive as far as the parameters are concerned. This manual uses lower case, unless the parameter in question is used only in legacy packages, or the context refers exclusively to such a package.

Serviceguard provides a set of parameters for configuring EMS (Event Monitoring Service) resources. These are *resource_name*, *resource_polling_interval*, *resource_start*, and *resource_up_value*. Configure each of these parameters in the package configuration file for each resource the package will be dependent on.

The *resource_start* parameter determines when Serviceguard starts up resource monitoring for EMS resources. *resource_start* can be set to either automaticor deferred.

Serviceguard will start up resource monitoring for automatic resources automatically when the Serviceguard cluster daemon starts up on the node.

Serviceguard will not attempt to start deferred resource monitoring during node startup, but will start monitoring these resources when the package runs.

The following is an example of how to configure deferred and automatic resources.

```
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/lan0
resource_polling_interval 60
resource_start automatic
resource_up_value = up
```

NOTE: For a legacy package, specify the deferred resources again in the package control script, using the *DEFERRED_RESOURCE_NAME* parameter:

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

If a resource is configured to be AUTOMATIC in a legacy configuration file, you do not need to define *DEFERRED_RESOURCE_NAME* in the package control script.

About Package Dependencies

Starting in Serviceguard A.11.17, a package can have dependencies on other packages, meaning the package will not start on a node unless the packages it depends on are running on that node.

In Serviceguard A.11.17, package dependencies are supported only for use with certain applications specified by HP, such as the multi-node and system multi-node packages that HP supplies for use with Veritas Cluster File System (CFS) on systems that support it.

As of Serviceguard A.11.18, package dependency is no longer restricted; you can make a package dependent on any other package or packages running on the same cluster node, subject to the restrictions spelled out in Chapter 6, under *dependency_condition* (page 228).

As of A.11.19, Serviceguard adds two new capabilities: you can specify broadly where the package depended on must be running, and you can specify that it must be down. These capabilities

are discussed later in this section under "Extended Dependencies" (page 132). You should read the next section, "Simple Dependencies" (page 129), first.

Simple Dependencies

A simple dependency occurs when one package requires another to be running on the same node. You define these conditions by means of the parameters *dependency_condition* and *dependency_location*, using the literal values UP and same_node, respectively. (For detailed configuration information, see the package parameter definitions starting with "*dependency_name*" (page 227). For a discussion of complex dependencies, see "Extended Dependencies" (page 132).)

Make a package dependent on another package if the first package cannot (or should not) function without the services provided by the second, on the same node. For example, pkg1 might run a real-time web interface to a database managed by pkg2 on the same node. In this case it might make sense to make pkg1 dependent on pkg2.

In considering whether or not to create a simple dependency between packages, use the Rules for Simple Dependencies and Guidelines for Simple Dependencies that follow.

Rules for Simple Dependencies

Assume that we want to make pkg1 depend on pkg2 .

NOTE: pkg1 can depend on more than one other package, and pkg2 can depend on another package or packages; we are assuming only two packages in order to make the rules as clear as possible.

- pkg1 will not start on any node unless pkg2 is running on that node.
- pkg1's *package_type* (page 223) and *failover_policy* (page 226) constrain the type and characteristics of pkg2, as follows:
  - If pkg1 is a multi-node package, pkg2 must be a multi-node or system multi-node package. (Note that system multi-node packages are not supported for general use.)
  - If pkg1 is a failover package and its *failover_policy* is min_package_node, pkg2 must be a multi-node or system multi-node package.
  - If pkg1 is a failover package and its *failover_policy* is configured_node, pkg2 must be:
    - a multi-node or system multi-node package, *or*
    - a failover package whose *failover_policy* is configured_node.
- pkg2 cannot be a failover package whose *failover_policy* is min_package_node.
- pkg2's node *node_name* list (page 223) must contain all of the nodes on pkg1's.
  - This means that if pkg1 is configured to run on any node in the cluster (*), pkg2 must also be configured to run on any node.

    NOTE: If pkg1 lists all the nodes, rather than using the asterisk (*), pkg2 must also list them.

  - Preferably the nodes should be listed in the same order if the dependency is between packages whose *failover_policy* is configured_node; cmcheckconf and cmapplyconf will warn you if they are not.
- A package cannot depend on itself, directly or indirectly.

  That is, not only must pkg1 not specify itself in the *dependency_condition* (page 228), but pkg1 must not specify a dependency on pkg2 if pkg2 depends on pkg1, or if pkg2 depends on pkg3 which depends on pkg1, etc.

- If pkg1 is a failover package and pkg2 is a multi-node or system multi-node package, and pkg2 fails, pkg1 will halt and fail over to the next node on its *node_name* list on which pkg2 is running (and any other dependencies, such as resource dependencies or a dependency on a third package, are met).
- In the case of failover packages with a configured_node *failover_policy*, a set of rules governs under what circumstances pkg1 can force pkg2 to start on a given node. This is called dragging and is determined by each package's *priority* (page 227). See "Dragging Rules for Simple Dependencies" (page 130).
- If pkg2 fails, Serviceguard will halt pkg1 and any other packages that depend directly or indirectly on pkg2.

  By default, Serviceguard halts packages in dependency order, the dependent package(s) first, then the package depended on. In our example, pkg1 would be halted first, then pkg2. If there were a third package, pkg3, that depended on pkg1, pkg3 would be halted first, then pkg1, then pkg2.

  If the halt script for any dependent package hangs, by default the package depended on will wait forever (pkg2 will wait forever for pkg1, and if there is a pkg3 that depends on pkg1, pkg1 will wait forever for pkg3). You can modify this behavior by means of the *successor_halt_timeout* parameter (page 225). (The successor of a package depends on that package; in our example, pkg1 is a successor of pkg2; conversely pkg2 can be referred to as a predecessor of pkg1.)

Dragging Rules for Simple Dependencies

The *priority* parameter (page 227) gives you a way to influence the startup, failover, and failback behavior of a set of failover packages that have a configured_node *failover_policy*, when one or more of those packages depend on another or others.

The broad rule is that a higher-priority package can drag a lower-priority package, forcing it to start on, or move to, a node that suits the higher-priority package.

NOTE: This applies only when the packages are automatically started (package switching enabled); cmrunpkg will never force a package to halt.

Keep in mind that you do not *have* to set *priority*, even when one or more packages depend on another. The default value, no_priority, may often result in the behavior you want. For example, if pkg1 depends on pkg2, and *priority* is set to no_priority for both packages, and other parameters such as *node_name* and *auto_run* are set as recommended in this section, then pkg1 will normally follow pkg2 to wherever both can run, and this is the common-sense (and may be the most desirable) outcome.

The following examples express the rules as they apply to two failover packages whose *failover_policy* (page 226) is configured_node. Assume pkg1 depends on pkg2, that node1, node2 and node3 are all specified (not necessarily in that order) under *node_name* (page 223) in the configuration file for each package, and that *failback_policy* (page 227) is set to automatic for each package.

NOTE: Keep the following in mind when reading the examples that follow, and when actually configuring priorities:

1. *auto_run* (page 224) should be set to `yes` for all the packages involved; the examples assume that it is.
2. Priorities express a ranking order, so a lower number means a higher priority (10 is a higher priority than 30, for example).

   HP recommends assigning values in increments of 20 so as to leave gaps in the sequence; otherwise you may have to shuffle all the existing priorities when assigning priority to a new package.

   `no_priority`, the default, is treated as a lower priority than any numerical value.
3. All packages with `no_priority` are by definition of equal priority, and there is no other way to assign equal priorities; a numerical priority must be unique within the cluster. See *priority* (page 227) for more information.

In a simple dependency, if pkg1 depends on pkg2, and pkg1's priority is lower than or equal to pkg2's, pkg2's node order dominates. Assuming `pkg2`'s node order is `node1`, `node2`, `node3`, then:

- On startup:
  - `pkg2` will start on `node1`, or `node2` if `node1` is not available or does not at present meet all of its dependencies, etc.
    - `pkg1` will start on whatever node `pkg2` has started on (no matter where that node appears on `pkg1`'s *node_name* list) provided all of `pkg1`'s other dependencies are met there.
    - If the node where `pkg2` has started does not meet all `pkg1`'s dependencies, `pkg1` will not start.
- On failover:
  - If `pkg2` fails on `node1`, `pkg2` will fail over to `node2` (or `node3` if `node2` is not available or does not currently meet all of its dependencies, etc.)
    - `pkg1` will fail over to whatever node `pkg2` has restarted on (no matter where that node appears on `pkg1`'s *node_name* list) provided all of `pkg1`'s dependencies are met there.
      - If the node where `pkg2` has restarted does not meet all `pkg1`'s dependencies, `pkg1` will not restart.
  - If `pkg1` fails, `pkg1` will not fail over.

    This is because `pkg1` cannot restart on any adoptive node until `pkg2` is running there, and `pkg2` is still running on the original node. `pkg1` cannot drag `pkg2` because it has insufficient priority to do so.
- On failback:
  - If both packages have moved from `node1` to `node2` and `node1` becomes available, `pkg2` will fail back to `node1` *only if `pkg2's` priority is higher than `pkg1's`*:
    - If the priorities are equal, neither package will fail back (unless `pkg1` is not running; in that case `pkg2` can fail back).
    - If `pkg2's` priority is higher than `pkg1`'s, `pkg2` will fail back to `node1`; `pkg1` will fail back to `node1` provided all of `pkg1`'s other dependencies are met there;
      - if `pkg2` has failed back to `node1` and `node1` does not meet all of `pkg1`'s dependencies, `pkg1` will halt.

In a simple dependency, if pkg1 depends on pkg2, and pkg1's priority is higher than pkg2's, pkg1's node order dominates. Assuming pkg1's node order is node1, node2, node3, then:

- On startup:
  - pkg1 will select node1 to start on, provided pkg2 can run there.
  - pkg2 will start on node1, provided it can run there (no matter where node1 appears on pkg2's *node_name* list).
    - If pkg2 is already running on another node, it will be dragged to node1, provided it can run there.
  - If pkg2 cannot start on node1, then both packages will attempt to start on node2 (and so on).

  Note that the nodes will be tried in the order of pkg1's *node_name* list, and pkg2 will be dragged to the first suitable node on that list whether or not it is currently running on another node.
- On failover:
  - If pkg1 fails on node1, pkg1 will select node2 to fail over to (or node3 if it can run there and node2 is not available or does not meet all of its dependencies; etc.)
  - pkg2 will be dragged to whatever node pkg1 has selected, and restart there; then pkg1 will restart there.
- On failback:
  - If both packages have moved to node2 and node1 becomes available, pkg1 will fail back to node1 if both packages can run there;
    - otherwise, neither package will fail back.

Guidelines for Simple Dependencies

As you can see from the "Dragging Rules for Simple Dependencies" (page 130), if pkg1 depends on pkg2, it can sometimes be a good idea to assign a higher priority to pkg1, because that provides the best chance for a successful failover (and failback) if pkg1 fails.

But you also need to weigh the relative importance of the packages. If pkg2 runs a database that is central to your business, you probably want it to run undisturbed, no matter what happens to application packages that depend on it. In this case, the database package should have the highest priority.

Note that, if no priorities are set, the dragging rules favor a package that is depended on over a package that depends on it.

Consider assigning a higher priority to a dependent package if it is about equal in real-world importance to the package it depends on; otherwise assign the higher priority to the more important package, or let the priorities of both packages default.

You also need to think about what happens when a package fails; see "What Happens when a Package Fails" (page 134).

Extended Dependencies

To the capabilities provided by Simple Dependencies (page 129), extended dependencies add the following:

- You can specify whether the package depended on must be running or must be down.

  You define this condition by means of the *dependency_condition*, using one of the literals UP or DOWN (the literals can be upper or lower case). We'll refer to the requirement that

another package be down as an exclusionary dependency; see "Rules for Exclusionary Dependencies" (page 133).

- You can specify where the *dependency_condition* must be satisfied: on the same node, a different node, all nodes, or any node in the cluster.

  You define this by means of the *dependency_location* parameter (page 228), using one of the literals `same_node`, `different_node`, `all_nodes`, or `any_node`.

  `different_node` and `any_node` are allowed only if *dependency_condition* is `UP`. `all_nodes` is allowed only if *dependency_condition* is `DOWN`.

  See "Rules for different_node and any_node Dependencies" (page 134).

For more information about the *dependency_* parameters, see the definitions starting with "*dependency_name*" (page 227), and the `cmmakepkg (1m)` manpage.

IMPORTANT: If you have not already done so, read the discussion of Simple Dependencies (page 129) before you go on.

The interaction of the legal values of *dependency_location* and *dependency_condition* creates the following possibilities:

- Same-node dependency: a package can require that another package be `UP` on the same node.

  This is the case covered in the section on Simple Dependencies (page 129).

- Different-node dependency: a package can require that another package be `UP` on a different node.
- Any-node dependency: a package can require that another package be `UP` on any node in the cluster.
- Same-node exclusion: a package can require that another package be `DOWN` on the same node. (But this does not prevent that package from being `UP` on another node.)
- All-nodes exclusion: a package can require that another package be `DOWN` on all nodes in the cluster.

Rules for Exclusionary Dependencies

- All exclusions must be mutual.

  That is, if `pkg1` requires `pkg2` to be `DOWN`, `pkg2` must also require `pkg1` to be `DOWN`.

  By creating an exclusionary relationship between any two packages, you ensure that only one of them can be running at any time — either on a given node (same-node exclusion) or throughout the cluster (all-nodes exclusion). A package can have an exclusionary relationship with any number of other packages, but each such relationship must be mutual.

  NOTE: Unexpected behavior may result if you simultaneously halt two packages that have an exclusionary dependency on each other.

- Priority (discussed in detail under "Dragging Rules for Simple Dependencies" (page 130)) must be set for at least one of the packages in an exclusionary relationship.

  The higher-priority package can force the lower-priority package to halt or (in the case of a same-node exclusion) move to another eligible node, if any.

- *dependency_location* must be either `same_node` or `all_nodes`, and must be the same for both packages.
- Both packages must be failover packages whose *failover_policy* (page 226) is `configured_node`.

Rules for different_node and any_node Dependencies

These rules apply to packages whose *dependency_condition* is UP and whose *dependency_location* is different_node or any_node. For same-node dependencies, see Simple Dependencies (page 129); for exclusionary dependencies, see "Rules for Exclusionary Dependencies" (page 133).

- Both packages must be failover packages whose *failover_policy* (page 226) is configured_node.
- The *priority* (page 227) of the package depended on must be higher than or equal to the priority of the dependent package and the priorities of that package's dependents.
  - For example, if pkg1 has a different_node or any_node dependency on pkg2, pkg2's priority must be higher than or equal to pkg1's priority and the priority of any package that depends on pkg1 to be UP. pkg2's node order dominates when Serviceguard is placing the packages.
- A package cannot depend on itself, directly or indirectly.

  For example, not only must pkg1 not specify itself in the *dependency_condition* (page 228), but pkg1 must not specify a dependency on pkg2 if pkg2 depends on pkg1, or if pkg2 depends on pkg3 which depends on pkg1, etc.
- "Dragging" rules apply. See "Dragging Rules for Simple Dependencies" (page 130).

What Happens when a Package Fails

This discussion applies to packages that have dependents, or are depended on, or both (UP dependencies only). When such a package fails, Serviceguard does the following:

1. Halts the packages that depend on the failing package, if any.

   Serviceguard halts the dependent packages (and any packages that depend on them, etc.) This happens regardless of the priority of the failed package.

   NOTE: Dependent packages are halted even in the case of different_node or any_node dependency. For example if pkg1 running on node1 has a different_node or any_node dependency on pkg2 running on node2, and pkg2 fails over to node3, pkg1 will be halted and restarted as described below.

   By default the packages are halted in the reverse of the order in which they were started; and if the halt script for any of the dependent packages hangs, the failed package will wait indefinitely to complete its own halt process. This provides the best chance for all the dependent packages to halt cleanly, but it may not be the behavior you want. You can change it by means of the *successor_halt_timeout* parameter (page 225). (A successor is a package that depends on another package.)

   If the failed package's *successor_halt_timeout* is set to zero, Serviceguard will halt the dependent packages in parallel with the failed package; if it is set to a positive number, Serviceguard will halt the packages in the reverse of the start order, but will allow the failed package to halt after the *successor_halt_timeout* number of seconds whether or not the dependent packages have completed their halt scripts.
2. Halts the failing package.

   After the successor halt timer has expired or the dependent packages have all halted, Serviceguard starts the halt script of the failing package, regardless of whether the dependents' halts succeeded, failed, or timed out.

3. Halts packages the failing package depends on, starting with the package this package immediately depends on. The packages are halted only if:
   - these are failover packages, *and*
   - the failing package can "drag" these packages to a node on which they can all run.

   Otherwise the failing package halts and the packages it depends on continue to run
4. Starts the packages the failed package depends on (those halted in step 3, if any).

   If the failed package has been able to drag the packages it depends on to the adoptive node, Serviceguard starts them in the reverse of the order it halted them in the previous step (that is, the package that does not depend on any other package is started first).
5. Starts the failed package.
6. Starts the packages that depend on the failed package (those halted in step 1).

For More Information

For more information, see:

- The parameter descriptions for *priority* (page 227) and *dependency_* (page 227), and the corresponding comments in the package configuration template file
- The `cmmakepkg (1m)` manpage
- The white paper *Serviceguard's Package Dependency Feature*, which you can find at http://www.hp.com/go/hpux-serviceguard-docs

About Package Weights

Package weights and node capacities allow you to restrict the number of packages that can run concurrently on a given node, or, alternatively, to limit the total package "weight" (in terms of resource consumption) that a node can bear.

For example, suppose you have a two-node cluster consisting of a large system and a smaller system. You want all your packages to be able to run on the large system at the same time, but, if the large node fails, you want only the critical packages to run on the smaller system. Package weights allow you to configure Serviceguard to enforce this behavior.

Package Weights and Node Capacities

You define a capacity, or capacities, for a node (in the cluster configuration file), and corresponding weights for packages (in the package configuration file).

Node capacity is consumed by package weights. Serviceguard ensures that the capacity limit you set for a node is never exceeded by the combined weight of packages running on it; if a node's available capacity will be exceeded by a package that wants to run on that node, the package will not run there. This means, for example, that a package cannot fail over to a node if that node does not currently have available capacity for it, even if the node is otherwise eligible to run the package — unless the package that wants to run has sufficient priority to force one of the packages that are currently running to move; see "How Package Weights Interact with Package Priorities and Dependencies" (page 141).

Configuring Weights and Capacities

You can configure multiple capacities for nodes, and multiple corresponding weights for packages, up to four capacity/weight pairs per cluster. This allows you considerable flexibility in managing package use of each node's resources — but it may be more flexibility than you need. For this reason Serviceguard provides two methods for configuring capacities and weights: a simple method and a comprehensive method. The subsections that follow explain each of these methods.

Simple Method

Use this method if you simply want to control the number of packages that can run on a given node at any given time. This method works best if all the packages consume about the same amount of computing resources.

If you need to make finer distinctions between packages in terms of their resource consumption, use the Comprehensive Method (page 137) instead.

To implement the simple method, use the reserved keyword `package_limit` to define each node's capacity. In this case, Serviceguard will allow you to define only this single type of capacity, and corresponding package weight, in this cluster. Defining package weight is optional; for `package_limit` it will default to 1 for all packages, unless you change it in the package configuration file.

Example 1

For example, to configure a node to run a maximum of ten packages at any one time, make the following entry under the node's *NODE_NAME* entry in the cluster configuration file:

```
NODE_NAME node1
. . .
CAPACITY_NAME package_limit
CAPACITY_VALUE 10
```

Now all packages will be considered equal in terms of their resource consumption, and this node will never run more than ten packages at one time. (You can change this behavior if you need to by modifying the weight for some or all packages, as the next example shows.) Next, define the *CAPACITY_NAME* and *CAPACITY_VALUE* parameters for the remaining nodes, setting *CAPACITY_NAME* to `package_limit` in each case. You may want to set *CAPACITY_VALUE* to different values for different nodes. A ten-package capacity might represent the most powerful node, for example, while the least powerful has a capacity of only two or three.

NOTE: Serviceguard does not require you to define a capacity for each node. If you define the *CAPACITY_NAME* and *CAPACITY_VALUE* parameters for some nodes but not for others, the nodes for which these parameters are not defined are assumed to have limitless capacity; in this case, those nodes would be able to run any number of eligible packages at any given time.

If some packages consume more resources than others, you can use the *weight_name* and *weight_value* parameters to override the default value (1) for some or all packages. For example, suppose you have three packages, `pkg1`, `pkg2`, and `pkg3`. `pkg2` is about twice as resource-intensive as `pkg3` which in turn is about one-and-a-half times as resource-intensive as `pkg1`. You could represent this in the package configuration files as follows:

- For `pkg1`:

  ```
  weight_name package_limit
  weight_value 2
  ```

- For `pkg2`:

  ```
  weight_name package_limit
  weight_value 6
  ```

- For `pkg3`:

  ```
  weight_name package_limit
  weight_value 3
  ```

Now `node1`, which has a *CAPACITY_VALUE* of 10 for the reserved *CAPACITY_NAME* `package_limit`, can run any two of the packages at one time, but not all three. If in addition

you wanted to ensure that the larger packages, pkg2 and pkg3, did not run on node1 at the same time, you could raise the *weight_value* of one or both so that the combination exceeded 10 (or reduce node1's capacity to 8).

Points to Keep in Mind

The following points apply specifically to the Simple Method (page 136). Read them in conjunction with the Rules and Guidelines (page 141), which apply to all weights and capacities.

- If you use the reserved *CAPACITY_NAME* package_limit, then this is the only type of capacity and weight you can define in this cluster.
- If you use the reserved *CAPACITY_NAME* package_limit, the default weight for all packages is 1. You can override this default in the *package* configuration file, via the *weight_name* and *weight_value* parameters, as in the example above.

  (The default weight remains 1 for any package to which you do not explicitly assign a different weight in the package configuration file.)
- If you use the reserved *CAPACITY_NAME* package_limit, *weight_name*, if used, must also be package_limit.
- You do not have to define a capacity for every node; if you don't, the node is assumed to have unlimited capacity and will be able to run any number of eligible packages at the same time.
- If you want to define only a single capacity, but you want the default weight to be zero rather than 1, do not use the reserved name package_limit. Use another name (for example resource_quantity) and follow the Comprehensive Method. This is also a good idea if you think you may want to use more than one capacity in the future.

To learn more about configuring weights and capacities, see the documents listed under For More Information (page 141).

Comprehensive Method

Use this method if the Simple Method (page 136) does not meet your needs. (Make sure you have read that section before you proceed.) The comprehensive method works best if packages consume differing amounts of computing resources, so that simple one-to-one comparisons between packages are not useful.

IMPORTANT: You cannot combine the two methods. If you use the reserved capacity package_limit for any node, Serviceguard will not allow you to define any other type of capacity and weight in this cluster; so you are restricted to the Simple Method in that case.

Defining Capacities

Begin by deciding what capacities you want to define; you can define up to four different capacities for the cluster.

You may want to choose names that have common-sense meanings, such as “processor”, “memory”, or “IO”, to identify the capacities, but you do not have to do so. In fact it could be misleading to identify single resources, such as “processor”, if packages really contend for sets of interacting resources that are hard to characterize with a single name. In any case, the real-world meanings of the names you assign to node capacities and package weights are outside the scope of Serviceguard. Serviceguard simply ensures that for each capacity configured for a node, the combined weight of packages currently running on that node does not exceed that capacity.

For example, if you define a *CAPACITY_NAME* and *weight_name* processor, and a *CAPACITY_NAME* and *weight_name* memory, and a node has a processor capacity of 10 and a memory capacity of 1000, Serviceguard ensures that the combined processor weight of packages running on the node at any one time does not exceed 10, and that the combined memory weight does not exceed 1000. But Serviceguard has no knowledge of the real-world

meanings of the names processor and memory; there is no mapping to actual processor and memory usage and you would get exactly the same results if you used the names apples and oranges, for example.

Suppose you have the following configuration:

- A two node cluster running four packages. These packages contend for resource we'll simply call A and B.
- node1 has a capacity of 80 for A and capacity of 50 for B.
- node2 has a capacity of 60 for A and capacity of 70 for B.
- pkg1 uses 60 of the A capacity and 15 of the B capacity.
- pkg2 uses 40 of the A capacity and 15 of the B capacity.
- pkg3 uses insignificant amount (zero) of the A capacity and 35 of the B capacity.
- pkg4 uses 20 of the A capacity and 40 of the B capacity.

pkg1 and pkg2 together require 100 of the A capacity and 30 of the B capacity. This means pkg1 and pkg2 cannot run together on either of the nodes. While both nodes have sufficient B capacity to run both packages at the same time, they do not have sufficient A capacity.

pkg3 and pkg4 together require 20 of the A capacity and 75 of the B capacity. This means pkg3 and pkg4 cannot run together on either of the nodes. While both nodes have sufficient A capacity to run both packages at the same time, they do not have sufficient B capacity.

Example 2

To define these capacities, and set limits for individual nodes, make entries such as the following in the cluster configuration file:

```
CLUSTER_NAME cluster_23
...
NODE_NAME node1
...
CAPACITY_NAME A
CAPACITY_VALUE 80
CAPACITY_NAME B
CAPACITY_VALUE 50
NODE_NAME node2
CAPACITY_NAME A
CAPACITY_VALUE 60
CAPACITY_NAME B
CAPACITY_VALUE 70
...
```

NOTE: You do not have to define capacities for every node in the cluster. If any capacity is not defined for any node, Serviceguard assumes that node has an infinite amount of that capacity. In our example, not defining capacity `A` for a given node would automatically mean that node could run `pkg1` and `pkg2` at the same time no matter what `A` weights you assign those packages; not defining capacity `B` would mean the node could run `pkg3` and `pkg4` at the same time; and not defining either one would mean the node could run all four packages simultaneously.

When you have defined the nodes' capacities, the next step is to configure the package weights; see "Defining Weights".

Defining Weights

Package weights correspond to node capacities, and for any capacity/weight pair, *`CAPACITY_NAME`* and *`weight_name`* must be identical.

You define weights for individual packages in the package configuration file, but you can also define a cluster-wide default value for a given weight, and, if you do, this default will specify the weight of all packages that do not explicitly override it in their package configuration file.

NOTE: There is one exception: system multi-node packages cannot have weight, so a cluster-wide default weight does not apply to them.

Defining Default Weights

To pursue the example begun under "Defining Capacities" (page 137), let's assume that all packages other than `pkg1` and `pkg2` use about the same amount of capacity `A`, and all packages other than `pkg3` and `pkg4` use about the same amount of capacity `B`. You can use the *`WEIGHT_DEFAULT`* parameter in the cluster configuration file to set defaults for both weights, as follows.

Example 3

```
WEIGHT_NAME A
WEIGHT_DEFAULT 20
WEIGHT_NAME B
WEIGHT_DEFAULT 15
```

This means that any package for which weight `A` is not defined in its package configuration file will have a weight `A` of 20, and any package for which weight `B` is not defined in its package configuration file will have a weight `B` of 15.

Given the capacities we defined in the cluster configuration file (see "Defining Capacities"), `node1` can run any three packages that use the default for both `A` and `B`. This would leave 20 units of spare `A` capacity on this node, and 5 units of spare `B` capacity.

Defining Weights for Individual Packages

For each capacity you define in the cluster configuration file (see "Defining Capacities") you have the following choices when it comes to assigning a corresponding weight to a given package:

1. Configure a cluster-wide default weight and let the package use that default.
2. Configure a cluster-wide default weight but override it for this package in its package configuration file.
3. Do not configure a cluster-wide default weight, but assign a weight to this package in its package configuration file.
4. Do not configure a cluster-wide default weight and do not assign a weight for this package in its package configuration file.

NOTE: Option 4 means that the package is "weightless" as far as this particular capacity is concerned, and can run even on a node on which this capacity is completely consumed by other packages.

(You can make a package "weightless" for a given capacity even if you have defined a cluster-wide default weight; simply set the corresponding weight to zero in the package's cluster configuration file.)

Pursuing the example started under "Defining Capacities" (page 137), we can now use options 1 and 2 to set weights for `pkg1` through `pkg4`.

Example 4

In `pkg1`'s package configuration file:

```
weight_name A
weight_value 60
```

In `pkg2`'s package configuration file:

```
weight_name A
weight_value 40
```

In `pkg3`'s package configuration file:

```
weight_name B
weight_value 35
weight_name A
weight_value 0
```

In `pkg4`'s package configuration file:

```
weight_name B
weight_value 40
```

IMPORTANT: *`weight_name`* in the package configuration file must exactly match the corresponding *`CAPACITY_NAME`* in the cluster configuration file. This applies to case as well as spelling: *`weight_name`* `a` would not match *`CAPACITY_NAME`* `A`.

You cannot define a weight unless the corresponding capacity is defined: `cmapplyconf` will fail if you define a weight in the package configuration file and no node in the package's *`node_name`* list (page 223) has specified a corresponding capacity in the cluster configuration file; or if you define a default weight in the cluster configuration file and no node in the cluster specifies a capacity of the same name.

Some points to notice about this example:

- Since we did not configure a `B` weight for `pkg1` or `pkg2`, these packages have the default `B` weight (15) that we set in the cluster configuration file in Example 3 (page 139). Similarly, `pkg4` has the default `A` weight (20).
- We have configured `pkg3` to have a `B` weight of 35, but no `A` weight.
- `pkg1` will consume all of `node2`'s `A` capacity; no other package that has `A` weight can run on this node while `pkg1` is running there.

  But `node2` could still run `pkg3` while running `pkg1`, because `pkg3` has no `A` weight, and `pkg1` is consuming only 15 units (the default) of `node2`'s `B` capacity, leaving 35 available to `pkg3` (assuming no other package that has `B` weight is already running there).
- Similarly, if any package that has `A` weight is already running on `node2`, `pkg1` will not be able to start there (unless `pkg1` has sufficient priority to force another package or packages to move; see "How Package Weights Interact with Package Priorities and Dependencies"

(page 141)). This is true whenever a package has a weight that exceeds the available amount of the corresponding capacity on the node.

Rules and Guidelines

The following rules and guidelines apply to both the Simple Method (page 136) and the Comprehensive Method (page 137) of configuring capacities and weights.

- You can define a maximum of four capacities, and corresponding weights, throughout the cluster.

  NOTE: But if you use the reserved *CAPACITY_NAME* package_limit, you can define only that single capacity and corresponding weight. See "Simple Method" (page 136).

- Node capacity is defined in the *cluster* configuration file, via the *CAPACITY_NAME* and *CAPACITY_VALUE* parameters.
- Capacities can be added, changed, and deleted while the cluster is running. This can cause some packages to be moved, or even halted and not restarted.
- Package weight can be defined in *cluster* configuration file, via the *WEIGHT_NAME* and *WEIGHT_DEFAULT* parameters, or in the *package* configuration file, via the *weight_name* and *weight_value* parameters, or both.
- Weights can be assigned (and *WEIGHT_DEFAULT*s, apply) only to multi-node packages and to failover packages whose *failover_policy* (page 226) is configured_node and whose *failback_policy* (page 227) is manual.
- If you define weight (*weight_name* and *weight_value*) for a package, make sure you define the corresponding capacity (*CAPACITY_NAME* and *CAPACITY_VALUE*) in the cluster configuration file for at least one node on the package's *node_name* list (page 223). Otherwise cmapplyconf will fail when you try to apply the package.
- Weights (both cluster-wide *WEIGHT_DEFAULT*s, and weights defined in the package configuration files) can be changed while the cluster is up and the packages are running. This can cause some packages to be moved, or even halted and not restarted.

For More Information

For more information about capacities, see the comments under *CAPACITY_NAME* and *CAPACITY_VALUE* in:

- the cluster configuration file
- the cmquerycl (1m) manpage
- the section "Cluster Configuration Parameters " (page 105) in this manual.

For more information about weights, see the comments under *weight_name* and *weight_value* in:

- the package configuration file
- the cmmakepkg (1m) manpage
- the section "Package Parameter Explanations" (page 222) in this manual.

For further discussion and use cases, see the white paper *Using Serviceguard's Node Capacity and Package Weight Feature* at http://www.hp.com/go/hpux-serviceguard-docs.

How Package Weights Interact with Package Priorities and Dependencies

If necessary, Serviceguard will halt a running lower-priority package that has weight to make room for a higher-priority package that has weight. But a running package that has no priority (that is, its *priority* is set to the default, no_priority) will not be halted to make room for a down package that has no priority. Between two down packages without priority, Serviceguard will

decide which package to start if it cannot start them both because there is not enough node capacity to support their weight.

Example 1

- pkg1 is configured to run on nodes turkey and griffon. It has a weight of 1 and a priority of 10. It is down and has switching disabled.
- pkg2 is configured to run on nodes turkey and griffon. It has a weight of 1 and a priority of 20. It is running on node turkey and has switching enabled.
- turkey and griffon can run one package each (package_limit is set to 1).

If you enable switching for pkg1, Serviceguard will halt the lower-priority pkg2 on turkey. It will then start pkg1 on turkey and restart pkg2 on griffon.

If neither pkg1 nor pkg2 had priority, pkg2 would continue running on turkey and pkg1 would run on griffon.

Example 2

- pkg1 is configured to run on nodes turkey and griffon. It has a weight of 1 and a priority of 10. It is running on node turkey and has switching enabled.
- pkg2 is configured to run on nodes turkey and griffon. It has a weight of 1 and a priority of 20. It is running on node turkey and has switching enabled.
- pkg3 is configured to run on nodes turkey and griffon. It has a weight of 1 and a priority of 30. It is down and has switching disabled.
- pkg3 has a same_node dependency on pkg2
- turkey and griffon can run two packages each (package_limit is set to 2).

If you enable switching for pkg3, it will stay down because pkg2, the package it depends on, is running on node turkey, which is already running two packages (its capacity limit). pkg3 has a lower priority than pkg2, so it cannot drag it to griffon where they both can run.

About External Scripts

The package configuration template for modular scripts explicitly provides for external scripts. These replace the CUSTOMER DEFINED FUNCTIONS in legacy scripts, and can be run either:

- On package startup and shutdown, as essentially the first and last functions the package performs. These scripts are invoked by means of the parameter *external_pre_script* (page 239); *or*
- During package execution, after volume-groups and file systems are activated, and IP addresses are assigned, and before the service and resource functions are executed; and again, in the reverse order, on package shutdown. These scripts are invoked by *external_script* (page 239).

The scripts are also run when the package is validated by cmcheckconf and cmapplyconf, and must have an entry point for validation; see below.

A package can make use of both kinds of script, and can launch more than one of each kind; in that case the scripts will be executed in the order they are listed in the package configuration file (and in the reverse order when the package shuts down).

In some cases you can rename or replace an external script while the package that uses it is running; see "Renaming or Replacing an External Script Used by a Running Package" (page 298).

Each external script must have three entry points: `start`, `stop`, and `validate`, and should exit with one of the following values:

- `0` - indicating success.
- `1` - indicating the package will be halted, and should not be restarted, as a result of failure in this script.
- `2` - indicating the package will be restarted on another node, or halted if no other node is available.

NOTE: In the case of the `validate` entry point, exit values `1` and `2` are treated the same; you can use either to indicate that validation failed.

The script can make use of a standard set of environment variables (including the package name, `SG_PACKAGE`, and the name of the local node, `SG_NODE`) exported by the package manager or the master control script that runs the package; and can also call a function to source in a logging function and other utility functions. One of these functions, `sg_source_pkg_env()`, provides access to all the parameters configured for this package, including package-specific environment variables configured via the *`pev_`* parameter (page 239).

NOTE: Some variables, including `SG_PACKAGE`, and `SG_NODE`, are available only at package run and halt time, not when the package is validated. You can use `SG_PACKAGE_NAME` at validation time as a substitute for `SG_PACKAGE`.

IMPORTANT: For more information, see the template in `$SGCONF/examples/external_script.template`.

A sample script follows. It assumes there is another script called `monitor.sh`, which will be configured as a Serviceguard service to monitor some application. The `monitor.sh` script (not included here) uses a parameter *`PEV_MONITORING_INTERVAL`*, defined in the package configuration file, to periodically poll the application it wants to monitor; for example:

```
PEV_MONITORING_INTERVAL 60
```

At validation time, the sample script makes sure the *`PEV_MONITORING_INTERVAL`* and the monitoring service are configured properly; at start and stop time it prints out the interval to the log file.

```
#!/bin/sh
# Source utility functions.
if [[ -z $SG_UTILS ]]
then
    . /etc/cmcluster.conf
    SG_UTILS=$SGCONF/scripts/mscripts/utils.sh
fi

if [[ -f ${SG_UTILS} ]]; then
    . ${SG_UTILS}
    if (( $? != 0 ))
    then
        echo "ERROR: Unable to source package utility functions file: ${SG_UTILS}"
        exit 1
    fi
else
    echo "ERROR: Unable to find package utility functions file: ${SG_UTILS}"
    exit 1
fi

# Get the environment for this package through utility function
# sg_source_pkg_env().
sg_source_pkg_env $*

function validate_command
{
    typeset -i ret=0
    typeset -i i=0
    typeset -i found=0
    # check PEV_ attribute is configured and within limits
    if [[ -z PEV_MONITORING_INTERVAL ]]
    then
        sg_log 0 "ERROR: PEV_MONITORING_INTERVAL attribute not configured!"
        ret=1
```

```
    elif (( PEV_MONITORING_INTERVAL < 1 ))
    then
        sg_log 0 "ERROR: PEV_MONITORING_INTERVAL value ($PEV_MONITORING_INTERVAL) not within legal limits!"
        ret=1
    fi
    # check monitoring service we are expecting for this package is configured
    while (( i < ${#SG_SERVICE_NAME[*]} ))
    do
        case ${SG_SERVICE_CMD[i]} in
            *monitor.sh*) # found our script
                          found=1
                          break
                          ;;
                       *)
                          ;;
        esac
        (( i = i + 1 ))
    done
    if (( found == 0 ))
    then
        sg_log 0 "ERROR: monitoring service not configured!"
        ret=1
    fi
    if (( ret == 1 ))
    then
        sg_log 0 "Script validation for $SG_PACKAGE_NAME failed!"
    fi
    return $ret
}
function start_command
{
    sg_log 5 "start_command"

    # log current PEV_MONITORING_INTERVAL value, PEV_ attribute can be changed
    # while the package is running

    sg_log 0 "PEV_MONITORING_INTERVAL for $SG_PACKAGE_NAME is $PEV_MONITORING_INTERVAL"
    return 0
}
function stop_command
{
    sg_log 5 "stop_command"
    # log current PEV_MONITORING_INTERVAL value, PEV_ attribute can be changed
    # while the package is running
    sg_log 0 "PEV_MONITORING_INTERVAL for $SG_PACKAGE_NAME is $PEV_MONITORING_INTERVAL"
    return 0
}
typeset -i exit_val=0
case ${1} in
     start)
           start_command $*
           exit_val=$?
           ;;
     stop)
           stop_command $*
           exit_val=$?
           ;;
     validate)
           validate_command $*
           exit_val=$?
           ;;
     *)
           sg_log 0 "Unknown entry point $1"
           ;;
esac
exit $exit_val
```

Using Serviceguard Commands in an External Script

You can use Serviceguard commands (such as `cmmodpkg`) in an external script run from a package. These commands must not interact with that package itself (that is, the package that runs the external script) but can interact with other packages. But be careful how you code these interactions.

If a Serviceguard command interacts with another package, be careful to avoid command loops. For instance, a command loop might occur under the following circumstances. Suppose a script run by `pkg1` does a `cmmodpkg -d` of `pkg2`, and a script run by `pkg2` does a `cmmodpkg -d` of `pkg1`. If both `pkg1` and `pkg2` start at the same time, the `pkg1` script now tries to `cmmodpkg pkg2`. But that `cmmodpkg` command has to wait for `pkg2` startup to complete. The `pkg2` script 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 specify a *`run_script_timeout`* and *`halt_script_timeout`* (page 225) for all packages, especially packages that use Serviceguard

commands in their external 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.

Determining Why a Package Has Shut Down

You can use an external script (or `CUSTOMER DEFINED FUNCTIONS` area of a legacy package control script) to find out why a package has shut down.

Serviceguard sets the environment variable *`SG_HALT_REASON`* in the package control script to one of the following values when the package halts:

- `failure` - set if the package halts because of the failure of a subnet, resource, or service it depends on
- `user_halt` - set if the package is halted by a `cmhaltpkg` or `cmhaltnode` command, or by corresponding actions in Serviceguard Manager
- `automatic_halt` - set if the package is failed over automatically because of the failure of a package it depends on, or is failed back to its primary node automatically (*`failback_policy`* = `automatic`)

You can add custom code to the package to interrogate this variable, determine why the package halted, and take appropriate action. For legacy packages, put the code in the `customer_defined_halt_cmds()` function in the `CUSTOMER DEFINED FUNCTIONS` area of the package control script; see "Adding Customer Defined Functions to the Package Control Script " (page 293). For modular packages, put the code in the package's external script; see "About External Scripts" (page 142).

For example, if a database package is being halted by an administrator (*`SG_HALT_REASON`* set to `user_halt`) you would probably want the custom code to perform an orderly shutdown of the database; on the other hand, a forced shutdown might be needed if *`SG_HALT_REASON`* is set to `failure, indicating that`the package is halting abnormally (for example because of the failure of a service it depends on).

last_halt_failed

`cmviewcl -v -f line` displays a `last_halt_failed` flag.

NOTE: `last_halt_failed` appears only in the line output of `cmviewcl`, not the default tabular format; you must use the `-v` and `-f line` options to see it.

The value of `last_halt_failed` is `no` if the halt script ran successfully, or was not run since the node joined the cluster, or was not run since the package was configured to run on the node; otherwise it is `yes`.

About Cross-Subnet Failover

It is possible to configure a cluster that spans subnets joined by a router, with some nodes using one subnet and some another. This is known as a cross-subnet configuration; see "Cross-Subnet Configurations" (page 29). In this context, you can configure packages to fail over from a node on one subnet to a node on another.

The implications for configuring a package for cross-subnet failover are as follows:

- For modular packages, you must configure two new parameters in the package configuration file to allow packages to fail over across subnets:
  - *`ip_subnet_node`* (page 231) - to indicate which nodes a subnet is configured on
  - *`monitored_subnet_access`* (page 230) - to indicate whether a monitored subnet is configured on all nodes (`FULL`) or only some (`PARTIAL`). (Leaving

monitored_subnet_access unconfigured for a monitored subnet is equivalent to FULL.)

(For legacy packages, see "Configuring Cross-Subnet Failover" (page 295).)

- You should not use the wildcard (*) for *node_name* (page 223) in the package configuration file, as this could allow the package to fail over across subnets when a node on the same subnet is eligible. Failing over across subnets can take longer than failing over on the same subnet. List the nodes in order of preference instead of using the wildcard.
- Each subnet interface (NIC) used by the package must have a standby interface on the local bridged net.

  The standby interface can be shared between subnets.
- Deploying applications in this environment requires careful consideration; see "Implications for Application Deployment" (page 146).
- If a *monitored_subnet* (page 229) is configured for PARTIAL *monitored_subnet_access* in a package's configuration file, it must be configured on at least one of the nodes on the *node_name* list for that package.

  Conversely, if all of the subnets that are being monitored for this package are configured for PARTIAL access, each node on the *node_name* list must have at least one of these subnets configured.

  - As in other cluster configurations, a package will not start on a node unless the subnets configured on that node, and specified in the package configuration file as monitored subnets, are up.

Implications for Application Deployment

Because the relocatable IP address will change when a package fails over to a node on another subnet, you need to make sure of the following:

- The hostname used by the package is correctly remapped to the new relocatable IP address.
- The application that the package runs must be configured so that the clients can reconnect to the package's new relocatable IP address.

  In the worst case (when the server where the application was running is down), the client may continue to retry the old IP address until TCP's *tcp_timeout* is reached (typically about ten minutes), at which point it will detect the failure and reset the connection.

For more information, see the white paper *Technical Considerations for Creating a Serviceguard Cluster that Spans Multiple IP Subnets*, at http://www.hp.com/go/hpux-serviceguard-docs.

Configuring a Package to Fail Over across Subnets: Example

To configure a package to fail over across subnets, you need to make some additional edits to the package configuration file.

NOTE: This section provides an example for a modular package; for legacy packages, see "Configuring Cross-Subnet Failover" (page 295).

Suppose that you want to configure a package, pkg1, so that it can fail over among all the nodes in a cluster comprising NodeA, NodeB, NodeC, and NodeD.

NodeA and NodeB use subnet 15.244.65.0, which is not used by NodeC and NodeD; and NodeC and NodeD use subnet 15.244.56.0, which is not used by NodeA and NodeB. (See "Obtaining Cross-Subnet Information" (page 181) for sample cmquerycl output).

Configuring node_name

First you need to make sure that pkg1 will fail over to a node on another subnet only if it has to. For example, if it is running on NodeA and needs to fail over, you want it to try NodeB, on the same subnet, before incurring the cross-subnet overhead of failing over to NodeC or NodeD.

Assuming nodeA is pkg1's primary node (where it normally starts), create *node_name* entries in the package configuration file as follows:

```
node_name nodeA
node_name nodeB
node_name nodeC
node_name nodeD
```

Configuring monitored_subnet_access

In order to monitor subnet 15.244.65.0 or 15.244.56.0, depending on where pkg1 is running, you would configure *monitored_subnet* and *monitored_subnet_access* in pkg1's package configuration file as follows:

```
monitored_subnet 15.244.65.0
monitored_subnet_access PARTIAL
monitored_subnet 15.244.56.0
monitored_subnet_access PARTIAL
```

NOTE: Configuring *monitored_subnet_access* as FULL (or not configuring *monitored_subnet_access*) for either of these subnets will cause the package configuration to fail, because neither subnet is available on all the nodes.

Configuring ip_subnet_node

Now you need to specify which subnet is configured on which nodes. In our example, you would do this by means of entries such as the following in the package configuration file:

```
ip_subnet 15.244.65.0
ip_subnet_node nodeA
ip_subnet_node nodeB
ip_address 15.244.65.82
ip_address 15.244.65.83
ip_subnet 15.244.56.0
ip_subnet_node nodeC
ip_subnet_node nodeD
ip_address 15.244.56.100
ip_address 15.244.56.101
```

Configuring a Package: Next Steps

When you are ready to start configuring a package, proceed to Chapter 6: "Configuring Packages and Their Services " (page 216); start with "Choosing Package Modules" (page 217). (If you find it helpful, you can assemble your package configuration data ahead of time on a separate worksheet for each package; blank worksheets are in Appendix E.)

Planning 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. For cross-subnet considerations, see "Cross-Subnet Configurations" (page 29).

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 *must 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. See "Cluster Lock Planning" (page 94) for more information.

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 (see "Cluster Configuration Parameters " is set high enough to accommodate the total number of packages you will be using.

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 tasks:

- Preparing Your Systems
- Configuring the Cluster (page 177)
- Managing the Running Cluster (page 211)

Configuring packages is described in the next chapter.

Preparing Your Systems

This section describes the tasks that should be done on the prospective cluster nodes before you actually configure the cluster. It covers the following topics:

- Installing and Updating Serviceguard
- Where Serviceguard Files Are Kept (page 149)
- Creating Cluster-wide Device Special Files (cDSFs) (page 150)
- Using Easy Deployment (page 152)
- Configuring Root-Level Access (page 157)
- Configuring Name Resolution (page 159)
- Ensuring Consistency of Kernel Configuration (page 161)
- Enabling the Network Time Protocol (page 162)
- Tuning Network and Kernel Parameters (page 162)
- Creating Mirrors of Root Logical Volumes (page 163)
- Choosing Cluster Lock Disks (page 164)
- Setting Up a Lock LUN (page 165)
- Setting Up and Running the Quorum Server (page 168)
- Creating the Storage Infrastructure and file systems with LVM, VxVM and CVM (page 168)

Installing and Updating Serviceguard

For information about installing Serviceguard, see the Release Notes for your version at http://www.hp.com/go/hpux-serviceguard-docs.

For information about installing and updating HP-UX, see the HP-UX Installation and Update Guide for the version you need: go to http://www.hp.com/go/hpux-core-docs and choose `HP-UX 11i v3`.

Appendix D (page 343) provides instructions for upgrading Serviceguard without halting the cluster. Make sure you read the entire Appendix, and the corresponding section in the Release Notes, before you begin.

Where Serviceguard Files Are Kept

Serviceguard uses a special file, `/etc/cmcluster.conf`, to define the locations for configuration and log files within the HP-UX file system. 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
SGRUN=/var/adm/cmcluster
SGAUTOSTART=/etc/rc.config.d/cmcluster
SGFFLOC=/opt/cmcluster/cmff
CMSNMPD_LOG_FILE=/var/adm/SGsnmpsuba.log
```

NOTE: If these variables are not defined on your system, then source the file `/etc/cmcluster.conf` in your login profile for user `root`. For example, you can add this line to `root`'s `.profile` file:

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

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

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

Creating Cluster-wide Device Special Files (cDSFs)

Cluster-wide device special files (cDSFs) are persistent device special files applied across a set of nodes. That is, they ensure that the same piece of storage has the same devicefile name on all of the nodes that share it; no matter how many paths there are to the device, the same cluster DSF is used to address it. If the device is moved, the same cDSF still addresses it.

For an overview, see "About Cluster-wide Device Special Files (cDSFs)" (page 99). For more information about persistent device special files, see "About Device File Names (Device Special Files)" (page 77).

Before You Start

Before you start, make sure you have read and understood the restrictions and limitations spelled out under "About Cluster-wide Device Special Files (cDSFs)" (page 99). You should also read the `cmsetdsfgroup (1m)` manpage.

If the cluster does not yet exist, make sure Serviceguard is installed on each prospective node.

Creating cDSFs for a Group of Nodes

Proceed as follows.

1. Identify the nodes for which you want to create cluster-wide device files; this is known as the cDSF group.

   This should be all the nodes in the cluster (or prospective cluster).

IMPORTANT: Remember that:

- You cannot create a cDSF group that crosses cluster boundaries; that is, the group must consist of the nodes of a single cluster.
- cDSFs use agile addressing; see "About Device File Names (Device Special Files)" (page 77) for information about agile addressing.

2. If the cluster does not yet exist, set up root access among the prospective nodes:
   a. If you have not already done so, set up `ssh` public/private key pairs on each node. This will allow the necessary commands to operate on all the prospective nodes before a cluster is formed.

      The simplest way to do this is via the DSAU `csshsetup` command; for example, if you are setting up a two-node cluster with nodes `node1` and `node2`, and you are logged in on `node1`:

      ```
      csshsetup -r node2
      ```

      For a large number of nodes, you might want to enter the node names into a file and use the `-f` option to get `csshsetup` to read the names from the file; for example, if you have stored the names in the file `/etc/cmcluster/sshhosts`:

      ```
      csshsetup -r -f /etc/cmcluster/sshhosts
      ```

      For more information about setting up `ssh` keys, see the *HP-UX Secure Shell Getting Started Guide* at http://www.hp.com/go/hpux-core-docs.

   b. Configure root access to each prospective node, using the hostname portion (only) of the fully-qualified domain name:

      ```
      cmpreparecl -n <node_name>  -n <node_name>  ...
      ```

      For example, for a cluster that will consist four nodes, `node1`, `node2`, `node3`, and `node4`:

      ```
      cmpreparecl -n node1 -n node2 -n node3 -n node4
      ```

      NOTE: Serviceguard must be installed on all of the nodes listed, and you must be logged in as superuser on one of these nodes to run the command.

3. Create the cDSFs.

NOTE: cDSFs apply only to shared storage; they will not be generated for local storage, such as root, boot, and swap devices.

- If the cluster does not exist yet, specify the name of each prospective node, for example:

  ```
  cmsetdsfgroup -n node1 -n node2 -n node3 -n node4
  ```

- If the cluster does exist, you can simply run:

  ```
  cmsetdsfgroup -c
  ```

  NOTE: You must be logged in as superuser on one of the cluster nodes. You do not need to provide the cluster name.

The cDSFs created by `cmsetdsfgroup` reside in `/dev/cdisk` for block device files and `/dev/rcdisk` for character devicefiles. You should use these new device files exclusively when you configure the cluster lock (if any) and package storage; see "Specifying a Lock Disk" (page 179),

"Specifying a Lock LUN" (page 180), and "Creating the Storage Infrastructure and file systems with LVM, VxVM and CVM" (page 168).

Adding a Node to a cDSF Group

When you add a new node to a cluster that uses cDSFs, you also need to add the node to the cDSF group. For example, to add the new cluster node `node5` to the existing cDSF group, run the following command from one of the existing members of the group:

```
cmsetdsfgroup -a -n node5
```

Removing a Node from a cDSF Group

When you remove a node from a cluster that uses cDSFs, you should also remove the node from the cDSF group. For example, to remove `node3` from the existing cDSF group, run the following command from one of the existing members of the group:

```
cmsetdsfgroup -r -n node3
```

Displaying the cDSF Configuration

- To see the members of a cDSF group, log in on one of the members and run `cmsetdsfgroup -q`
- To display all the information about the configuration stored in the configuration file `/etc/cmcluster/cdsf/cdsf_config`, log in on one of the cDSF group members and run `cmsetdsfgroup -v -q`

△ **CAUTION:** Do not modify `cdsf_config`.

- To report information (in line output format only) about DSFs, cDSFs, and volume groups for one or more nodes, use `cmquerystg (1m)`.
- You can also use the HP-UX command `io_cdsf_config (1m)` to display information about cDSFs.

See the manpages for more information.

Migrating Existing LVM Cluster Storage to cDSFs

You can migrate existing LVM cluster storage to cDSFs using the HP-UX command `vgcdsf (1m)`. See the manpage for more information. See also "LVM Commands and cDSFs" (page 100).

IMPORTANT: If you change the cluster lock volume or volumes to cDSFs, you need to change the cluster lock information in the cluster configuration file and re-apply the configuration; follow one of the procedures under "Updating the Cluster Lock Configuration" (page 281).

Using Easy Deployment

Easy Deployment commands enable you to get a cluster up and running in the minimum amount of time. Instructions follow. For an overview, see "About Easy Deployment" (page 100).

Before You Start

IMPORTANT: Before you start, you should have done the planning and preparation outlined in Chapter 4 (page 88). You must also do the following.

- Install Serviceguard on each node that is to be configured into the cluster; see "Installing and Updating Serviceguard " (page 149).

  You must have superuser capability on each node.
- Make sure all the nodes have access to at least one fully configured network.

- Make sure all the subnets used by the prospective nodes are accessible to *all* the nodes.

  Easy Deployment commands will fail if they detect an asymmetric network configuration, in which only a subset of nodes has access to a given subnet.

- Make sure that the LAN or LANs that will be used for the cluster heartbeat meet high-availability requirements.

  See the requirements for *HEARTBEAT_IP* listed under "Cluster Configuration Parameters " (page 105), and the networking "Rules and Restrictions" (page 27). In addition, HP recommends that you do not rely only on the site LAN, or any busy LAN, for the cluster heartbeat.

  For a discussion of cluster networking topology, see "Redundant Ethernet Configuration " (page 28).

  NOTE: You cannot use the Easy Deployment commands to create a cross-subnet configuration, as described under "Cross-Subnet Configurations" (page 29).

- If you have not already done so, set up `ssh` public/private key pairs on each node.

  This will allow the Easy Deployment commands to operate on all the prospective nodes before cluster is formed. If you need to set up the keys, you can use DSAU to simplify the task. For more information about setting up `ssh` keys, see the *HP-UX Secure Shell Getting Started Guide* at http://www.hp.com/go/hpux-core-docs.

- If you will be using a lock LUN (as opposed to an LVM lock disk or a quorum server) set up the LUN partition before running `cmdeploycl`; see "Setting Up a Lock LUN" (page 165).
- If you will be using a quorum server, it must be running and reachable from all the configured nodes through all the quorum server IP addresses before running `cmdeploycl`; see "Specifying a Quorum Server" (page 180).
- If you will be using individual disks (as opposed to RAID) for package data, and you will be mirroring them with MirrorDisk/UX as HP recommends, make sure the hardware is configured so as to allow PVG-strict mirroring.

  For more information, see "Using Mirrored Individual Data Disks" (page 169) — and it is good idea to read the entire section on "Creating a Storage Infrastructure with LVM" (page 168) so that you understand what `cmpreparestg` is doing.

- Make sure you have read and understood the additional limitations spelled out under "About Easy Deployment" (page 100); and it is a good idea to read the Easy Deployment manpages (`cmpreparecl (1m)`, `cmdeploycl (1m)`, and `cmpreparestg (1m)`) as well.

Using Easy Deployment Commands to Configure the Cluster

The examples that follow configure a two-node cluster with an LVM lock disk and a logical volume (with PVG-strict mirroring) for use by packages. For more information, and other options, see the manpages for `cmpreparecl (1m)`, `cmdeploycl (1m)`, and `cmpreparestg (1m)`. You can also use easy deployment commands to deploy a VxVM/CVM disk group. The following example illustrates its use with LVM storage.

NOTE: The actions taken by the commands you will be using are logged to `/var/adm/cmcluster/sgeasy/easy_deployment.log`, providing a useful record of the changes to system files.

1. Obtain networking information for the nodes on which you want the cluster heartbeat:

   `cmquerycl -N <network_template_file> -n <node1> -n <node2>`

   For example:

   `cmquerycl -N $SGCONF/mynetwork`

2. Edit the output file (`$SGCONF/mynetwork` in this example) to add entries for additional heartbeats, if necessary.

   NOTE: For information about heartbeat and networking requirements, see the sections listed under "Before You Start" (page 152).

   If you omit steps 1 and 2, and all the prospective nodes are connected to at least one subnet, `cmdeploycl` behaves as follows (the configuration is actually done by `cmquerycl (1m)` which is called by `cmdeploycl`).

   - If multiple subnets are configured among the nodes, `cmdeploycl` chooses a subnet with standby interfaces as the heartbeat.
   - If multiple subnets are configured, but no subnet has standby interfaces, `cmdeploycl` chooses two subnets for the heartbeat.
   - If only one subnet is configured, `cmdeploycl` configures that subnet as the heartbeat.

   △ **CAUTION:** If the subnet has no standby interfaces, the cluster will run, but will not meet high-availability requirements for the heartbeat. You must reconfigure the heartbeat as soon as possible; see "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).

3. If you have not already done so, create cluster-wide device special files (cDSFs).

   This step is optional, but HP strongly recommends it. For instructions, see "Creating Cluster-wide Device Special Files (cDSFs)" (page 150).

4. Create and start the cluster, configuring security and networking files, creating and deploying shared storage for an LVM lock disk:

   `cmdeploycl -c <clustername> -n <node1> -n <node2> -N <network_template_file> -b -L <vg>:<pv>`

   `<clustername>` must be the unique name for this cluster. *`<node1>`* and *`<node2>`* must be the hostname portion, and *only* the hostname portion, of each node's fully-qualified domain name (see "Configuring Name Resolution" (page 159) for more information about node names and hostnames). *`<network_template_file>`* is required when you need to configure networks on the specified nodes; however, this is not mandatory for cluster deployment. *`<vg>`*:*`<pv>`*, if used, must be, respectively, a volume group name that does not already exist, and the name of a physical volume that is unused and is not configured into any volume manager. HP recommends you use cluster-wide device special files (cDSFs); see "About Cluster-wide Device Special Files (cDSFs)" (page 99). In any case, the device special file (DSF) must use the agile addressing convention; see "About Device File Names (Device Special Files)" (page 77).

   For example:

   `cmdeploycl -c cluster1 -n ftsys9 -n ftsys10 -N $SGCONF/mynetwork -b -L /dev/vglock:/dev/cdsk/disk13`

   This command in this example does the following:

   - Calls `cmapplyconf (1m)` to configure the heartbeat subnets specified in `$SGCONF/mynetwork`
   - Updates `$SGCONF/cmclnodelist`, `/etc/hosts`, `/etc/nsswitch.conf`, and `/etc/inetd.conf` on each node.

NOTE: You can also accomplish this by running `cmpreparecl`; `cmdeploycl` calls `cmpreparecl`.

For more information, see "Configuring Root-Level Access" (page 157) and "Configuring Name Resolution" (page 159). Those sections describe the "manual" (HP-UX command-line) method of accomplishing what `cmpreparecl` does.

- Configures a two-node cluster (`cluster1`).
- Configures the LVM volume group and physical volume for the lock disk.
- Starts the cluster on `ftsys9` and `ftsys10`.

NOTE: The cluster does not yet have shared storage for packages.

Other forms of `cmdeploycl` allow you to create the cluster with a quorum server or lock LUN instead of a lock disk; see the manpage for more information.

If you use a quorum server, the quorum server software must already be installed and configured on the quorum server; see the latest version of the *HP Serviceguard Quorum Server Version A.04.00 Release Notes*, at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard Quorum Server Software`. `cmdeploycl` requires an IPv4 address (or addresses) for the quorum server; to configure IPv6 addresses, see "Specifying a Quorum Server" (page 180).

If shared storage already exists on both nodes, and you use `cmdeploycl` without any `-L` option, `cmdeploycl` will create an LVM lock disk using the shared storage.

See `cmdeploycl (1m)` for more information.

5. Create shared storage for a LVM volume group with mirroring for use by cluster packages.

   cmpreparestg -l *<vgname>* -p *<pv_path>* ... -n *<node1>* -n *<node2>*

   <vgname> is the path name of the volume group that is to be created and imported to each specified node, and each *<pv_path>* is the name of a physical volume on the local node from which the volume group specified by *<vgname>* is to be created.

   For example, suppose that /dev/cdisk/disk12 and /dev/cdisk/disk13 are two physical devices on the primary SCSI bus or adapter, and /dev/cdisk/disk14 and /dev/cdisk/disk15 are two physical devices on a secondary bus or adapter, and you want the devices on the secondary bus to contain PVG-strict mirror copies, proceed as follows.

   a. **cmpreparestg -l /dev/vgdatabase -p /dev/cdisk/disk12 -p /dev/cdisk/disk13 -p /dev/cdisk/disk14 -p /dev/cdisk/disk15 -n ftsys9 -n ftsys10**

      This configures physical devices /dev/cdisk/disk12 and /dev/cdisk/disk13 on the primary bus, and /dev/cdisk/disk14 and /dev/cdisk/disk15 on the secondary bus, into the volume group /dev/vgdatabase, and imports the volume group on the non-local node in the cluster created in step b.

   b. Edit /etc/lvmpvg on node1 and node2 so the relevant portion now reads as follows:

```
VG     /dev/vgdatabase
PVG    bus0
/dev/cdisk/disk12
/dev/cdisk/disk13

PVG    bus1
/dev/cdisk/disk14
/dev/cdisk/disk15
```

      NOTE: If you are not using cDSFs, you may need to change the DSF names to make sure that the DSF names point to the same physical storage on each node. If you are using cDSFs, the names are guaranteed to be the same.

   c. **cmpreparestg -l /dev/vgdatabase -L -o lv_opts="-L 120 -m 1 -s g" -o fs_opts="-F vxfs" -m /newdir -n ftsys9 -n ftsys10**

   This configures the logical volume /dev/vgdatabase/lvol*<n>* with PVG-strict mirroring from the disks on the primary bus (bus0, disks /dev/cdisk/disk12 and /dev/cdisk/disk13) to those on the secondary bus (bus1, disks /dev/cdisk/disk14 and /dev/cdisk/disk15). LVM supplies the logical volume number (*<n>*).

   This command also builds a VxFS file system on the logical volume, and mounts it to /newdir. If the logical volume will be used for raw storage (for example for a database) omit the fsopts and –m options; for example:

   cmpreparestg -l /dev/vgdatabase -L -o lv_opts="-L 120 -s g" -n ftsys9 -n ftsys10

   You can also create a shared storage for a VxVM/CVM disk group for use by cluster packages. For example, suppose that /dev/disk/disk39 and /dev/disk/disk40 are two physical devices and you want to create a VxVM/CVM disk group and configure a logical volume with mirroring across the two disks, use the cmpreparestg command as follows:

   cmpreparestg -g dg_sat -p /dev/disk/disk39 -p /dev/disk/disk40 -L lvol1 -o lv_opts="100M layout=mirror nmirror=2" -o fs_opts="-F vxfs" -m /mnt1

   This command creates a VxVM/CVM disk group with name dg_sat and create a logical volume lvol1 with mirroring across the two disks /dev/disk/disk39 and /dev/disk/disk40.

6. Verify the storage using the following commands:
   - For LVM volume groups:

     vgdisplay
   - For VxVM/CVM disk groups:

     vxdg list

The volume group or the disk group can now be used by packages; see Chapter 6 (page 216).

NOTE: `cmpreparestg` requires either `-l <vgname>` or `-g <dgname>` in all cases. The command must be run on one of the specified nodes (`ftsys9` or `ftsys10` in this example).

Other forms of `cmpreparestg` allow you to import a volume group or disk group that already exists, create a single logical volume or multiple logical volumes, and create and mount a file system on it. You can use the `-o vg_opts` option to specify options for newly created volume groups; this is particularly useful if you need to create an LVM version 2.`<x>` volume group. You can also use the `-o dg_opts` option to specify options for newly created VxVM/CVM disk groups. See the manpage for more information.

Configuring Root-Level Access

The subsections that follow explain how to set up HP-UX root access between the nodes in the prospective cluster. (When you proceed to configuring the cluster, you will define various levels of non-root access as well; see "Controlling Access to the Cluster" (page 183).)

As of Serviceguard A.11.20, much of this configuration can be done by means of the Serviceguard command `cmpreparecl (1m)` (or `cmpdeploycl (1m)`, which calls `cmpreparecl`). See the `cmpreparecl` manpage for more information. See also "Using Easy Deployment Commands to Configure the Cluster" (page 153).

NOTE: For more information and advice, see the white paper *Securing Serviceguard* at http://www.hp.com/go/hpux-serviceguard-docs.

Allowing Root Access to an Unconfigured Node

To enable a system to be included in a cluster, you must enable HP-UX root access to the system by the root user of every other potential cluster node. The Serviceguard mechanism for doing this is the file `$SGCONF/cmclnodelist`. This is sometimes referred to as a "bootstrap" file because Serviceguard consults it only when configuring a node into a cluster for the first time; it is ignored after that. It does not exist by default, but you will need to create it.

NOTE: `cmclnodelist` is created automatically by `cmpreparecl (1m)` (or `cmpdeploycl (1m)`, which calls `cmpreparecl`). You can skip the rest of this subsection if you have used, or plan to use, either of these commands. See "Using Easy Deployment Commands to Configure the Cluster" (page 153).

You may want to add a comment such as the following at the top of the file:

```
##########################################################
# Do not edit this file!
# Serviceguard uses this file only to authorize access to an
# unconfigured node. Once the node is configured,
# Serviceguard will not consult this file.
##########################################################
```

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

<hostname> <user> #<comment>

For example:

```
gryf   root      #cluster1, node1
sly    root      #cluster1, node2
bit    root      #cluster1, node3
```

This example grants root access to the node on which this `cmclnodelist` file resides to root users on the nodes `gryf`, `sly`, and `bit`.

Serviceguard also accepts the use of a "+" in the `cmclnodelist` file; this indicates that the root user on any Serviceguard node can configure Serviceguard on this node.

IMPORTANT: If `$SGCONF/cmclnodelist` does not exist, Serviceguard will look at `~/.rhosts`. HP strongly recommends that you use `cmclnodelist`.

NOTE: When you upgrade a cluster from Version A.11.15 or earlier, entries in `$SGCONF/cmclnodelist` are automatically updated to Access Control Policies in the cluster configuration file. All non-root user-hostname pairs are assigned the role of Monitor.

Ensuring that the Root User on Another Node Is Recognized

The HP-UX root user on any cluster node can configure the cluster. This requires that Serviceguard on one node be able to recognize the root user on another.

Serviceguard uses the `identd` daemon to verify user names, and, in the case of a root user, verification succeeds only if `identd` returns the username `root`. Because `identd` may return the username for the first match on UID 0, you must check `/etc/passwd` on each node you intend to configure into the cluster, and ensure that the entry for the root user comes before any other entry with a UID of 0.

NOTE: You need to do this even if you plan to use `cmpreparecl (1m)` or `cmpdeploycl (1m)`, which calls `cmpreparecl`. For more information about these commands, see "Using Easy Deployment Commands to Configure the Cluster" (page 153).

About identd

HP strongly recommends that you use `identd` for user verification, so you should make sure that each prospective cluster node is configured to run it. `identd` is usually started by `inetd` from `/etc/inetd.conf`.

NOTE: If you plan to use `cmpreparecl (1m)` (or `cmpdeploycl (1m)`, which calls `cmpreparecl`), you can skip the rest of this subsection.

Make sure that a line such as the following is uncommented in `/etc/inetd.conf`:

```
auth stream tcp6 wait bin /usr/lbin/identd identd
```

NOTE: If the `-t` option to `identd` is available on your system, you should set it to 120 (`-t120`); this ensures that a connection inadvertently left open will be closed after two minutes. In this case, the `identd` entry in `/etc/inetd.conf` should look like this:

```
auth stream tcp6 wait bin /usr/lbin/identd identd -t120
```

Check the man page for `identd` to determine whether the `-t` option is supported for your version of `identd`

(It is possible to disable `identd`, though HP recommends against doing so. If for some reason you have to disable `identd`, see "Disabling identd" (page 214).)

For more information about `identd`, see the white paper *Securing Serviceguard* at http://www.hp.com/go/hpux-serviceguard-docs, and the `identd (1M)` manpage.

Configuring Name Resolution

Serviceguard uses the name resolution services built into HP-UX.

NOTE: If you plan to use `cmpreparecl (1m)` (or `cmpdeploycl (1m)`, which calls `cmpreparecl`), the `/etc/hosts` and `/etc/nsswitch.conf` configuration described in this subsection will be done automatically, but you should still read the entire subsection and make sure you understand the issues.

In particular, you still need to make sure that aliases are properly represented in `/etc/hosts`, as described below.

Serviceguard nodes can communicate over any of the cluster's shared networks, so the network resolution service you are using (such as DNS, NIS, or LDAP) must be able to resolve each of their primary addresses on each of those networks to the primary hostname of the node in question.

In addition, HP recommends that you define name resolution in each node's `/etc/hosts` file, rather than rely solely on a service such as DNS. Configure the name service switch to consult the `/etc/hosts` file before other services. See "Safeguarding against Loss of Name Resolution Services" (page 160) for instructions.

NOTE: If you are using private IP addresses for communication within the cluster, and these addresses are not known to DNS (or the name resolution service you use) these addresses *must* be listed in `/etc/hosts`.

For requirements and restrictions that apply to IPv6–only clusters and mixed-mode clusters, see "Rules and Restrictions for IPv6-Only Mode" (page 103) and "Rules and Restrictions for Mixed Mode" (page 104), respectively, and the latest version of the Serviceguard release notes.

Keep the following rules in mind when creating entries in a Serviceguard node's`/etc/hosts`:

- *`NODE_NAME`* in the cluster configuration file must be identical to the hostname which is the first element of a fully qualified domain name (a name with four elements separated by periods). This hostname is what is returned by the HP-UX `hostname(1)` command.

  For example, the *`NODE_NAME`* should be `gryf` rather than `gryf.uksr.hp.com`. For more information, see the *`NODE_NAME`* entry under "Cluster Configuration Parameters " (page 105).

  NOTE: Since Serviceguard recognizes only the hostname, `gryf.uksr.hp.com` and `gryf.cup.hp.com` cannot be nodes in the same cluster, as Serviceguard would see them as the same host `gryf`.

- All primary IP addresses configured on the system must map to an entry in which the *`NODE_NAME`* is *either:*
  - the official hostname, as defined by `hosts (4)`, for example

    ```
    15.145.162.131   gryf
    ```

    or :

  - any of the aliases. Examples:

    ```
    15.145.162.131  gryf.uksr.hp.com   gryf
    10.8.0.131      gryf.uksr.hp.com   gryf
    10.8.1.131      gryf.uksr.hp.com   gryf

    15.145.162.132  sly.uksr.hp.com    sly
    10.8.0.132      sly.uksr.hp.com    sly
    10.8.1.132      sly.uksr.hp.com    sly

    ...
    ```

```
15.145.162.131  gryf.uksr.hp.com   gryf
10.8.0.131      gryf2.uksr.hp.com  gryf
10.8.1.131      gryf3.uksr.hp.com  gryf

15.145.162.132  sly.uksr.hp.com    sly
10.8.0.132      sly2.uksr.hp.com   sly
10.8.1.132      sly3.uksr.hp.com   sly
```

If applications require the use of hostname aliases, the Serviceguard hostname must be one of the aliases in all the entries for that host. For example, if the two-node cluster in the previous example were configured to use the aliases `alias-node1` and `alias-node2`, then the entries in `/etc/hosts` should look something like this:

```
15.145.162.131  gryf.uksr.hp.com   gryf  alias-node1
10.8.0.131      gryf2.uksr.hp.com  gryf  alias-node1
10.8.1.131      gryf3.uksr.hp.com  gryf  alias-node1

15.145.162.132  sly.uksr.hp.com    sly   alias-node2
10.8.0.132      sly2.uksr.hp.com   sly   alias-node2
10.8.1.132      sly3.uksr.hp.com   sly   alias-node2
```

IMPORTANT: Serviceguard does not support aliases for IPv6 addresses.

For information about configuring an IPv6–only cluster, or a cluster that uses a combination of IPv6 and IPv4 addresses for the nodes' hostnames, see "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102).

Safeguarding against Loss of Name Resolution Services

When you employ any user-level Serviceguard command (including `cmviewcl`), the command uses the name service you have configured (such as DNS) to obtain the addresses of all the cluster nodes. If the name service is not available, the command could hang or return an unexpected networking error message.

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 corresponding Serviceguard Manager functions) are affected, not the cluster daemon or package services.

The procedure that follows shows how to create a robust name-resolution configuration that will allow cluster nodes to continue communicating with one another if a name service fails. If a standby LAN is configured, this approach also allows the cluster to continue to function fully (including commands such as `cmrunnode` and `cmruncl`) after the primary LAN has failed.

NOTE: If a NIC fails, the affected node will be able to fail over to a standby LAN so long as the node is running in the cluster. But if a NIC that is used by Serviceguard fails when the affected node is not running in the cluster, Serviceguard will not be able to restart the node. (For instructions on replacing a failed NIC, see "Replacing LAN or Fibre Channel Cards" (page 314).)

NOTE: If you plan to use `cmpreparecl (1m)` (or`cmpdeploycl (1m)`, which calls `cmpreparecl`), the `/etc/hosts` and `/etc/nsswitch.conf`configuration described the procedure that follows will be done automatically, but you should still read the entire subsection and make sure you understand the issues.

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; see "Configuring Name Resolution" (page 159) for discussion and examples.

   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. If you are using DNS, make sure your name servers are configured in `/etc/resolv.conf`, for example:

   ```
   domain cup.hp.com
   search cup.hp.com hp.com
   nameserver 15.243.128.51
   nameserver 15.243.160.51
   ```

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

   - for DNS, enter (two lines):

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

   - for NIS, enter (two lines):

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

   If a line beginning with the string `hosts:` or `ipnodes:` already exists, then make sure that the text immediately to the right of this string is (on one line):

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

   or

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

   This step is critical, allowing the cluster nodes to resolve hostnames to IP addresses while DNS, NIS, or the primary LAN is down.

4. Create a `$SGCONF/cmclnodelist` file on all nodes that you intend to configure into the cluster, and allow access by all cluster nodes. See "Allowing Root Access to an Unconfigured Node" (page 157).

NOTE: HP recommends that you also make the name service itself highly available, either by using multiple name servers or by configuring the name service into a Serviceguard package.

Ensuring Consistency of Kernel Configuration

Make sure that the HP-UX 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

HP strongly recommends 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 HP-UX manual *HP-UX Internet Services Administrator's Guide* posted at http://www.hp.com/go/hpux-networking-docs.

Tuning Network and Kernel Parameters

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

You may need to adjust these parameters for larger cluster configurations and applications.

- `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)`.

Make adjustments with care, and if you experience problems, return the parameters to their default values.

NOTE: If you contact HP support regarding Serviceguard or networking, please be sure to mention any parameters that have been changed from their defaults.

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

- *`ip6_nd_dad_solicit_count`* — This network parameter enables the Duplicate Address Detection feature for IPv6 addresses. You can find more information l under "IPv6 Relocatable Address and Duplicate Address Detection Feature" (page 367).
- *`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.

  The following requirements must 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.
- *`ip_strong_es_model`* — This network parameter controls support for the strong end-system model, which prevents a system from acting as a router. For more information about this

model, see the *HP-UX IPSec Version A.03.00 Administrator's Guide* which you can find at http://www.hp.com/go/hpux-security-docs —> HP-UX IPSec Software.

ip_strong_es_model is disabled (set to 0) by default. Setting it to 1 enables it.

CAUTION: HP supports enabling this parameter only if you are *not* using a cross-subnet configuration (page 29). Otherwise, leave the parameter at its default setting (zero, meaning disabled).

Enabling this parameter allows you to configure a default gateway for each physical IPv4 interface. These gateways allow a system to send an unbound packet through the interface for the address to which the socket (or communication endpoint) is bound. If the socket (or communication endpoint) is not bound to a specific address, the system sends the packet through the interface on which the unbound packet was received.

This means that the packet source addresses (and therefore the interfaces on a multihomed host) affect the selection of a gateway for outbound packets once *ip_strong_es_model* is enabled. For more information see "Using a Relocatable Address as the Source Address for an Application that is Bound to INADDR_ANY" (page 335).

Creating Mirrors of Root Logical Volumes

HP strongly recommends 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.

IMPORTANT: This must be done, as described below, whether or not you intend to use cmpreparestg (1m) to configure storage. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information about cmpreparestg.

NOTE: Under agile addressing, the physical devices in these examples would have names such as /dev/[r]disk/disk1, and /dev/[r]disk/disk2. See "About Device File Names (Device Special Files)" (page 77).

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 those for /usr, /swap, 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. See “Cluster Lock ” (page 44) and “Cluster Lock Planning” (page 94) for discussion of cluster lock options.

The cluster lock disk is configured on an LVM 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 I/O timeout values are used for the cluster lock physical volumes. Changing the I/O 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 I/O 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.” For instructions on setting up a lock disk, see “Specifying a Lock Disk” (page 179).

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.

Setting Up a Lock LUN

LUN stands for Logical Unit Number. The term can refer to a single physical disk, but these days is more often used in a SAN (Storage Area Network) or NAS (Network-Attached Storage) context to denote a virtual entity derived from one or more physical disks.

Keep the following points in mind when choosing a device for a lock LUN:

- All the cluster nodes must be physically connected to the lock LUN.
- A lock LUN must be a block device.
- All existing data on the LUN will be destroyed when you configure it as a lock LUN.

  This means that if you use an existing lock disk, the existing lock information will be lost, and if you use a LUN that was previously used as a lock LUN for a Linux cluster, that lock information will also be lost.
- A lock LUN cannot also be used in an LVM physical volume or VxVM or CVM disk group.
- A lock LUN cannot be shared by more than one cluster.
- A lock LUN cannot be used in a dual-lock configuration.
- You do not need to back up the lock LUN data, and in fact there is no way to do so.

A lock LUN needs only a small amount of storage, about 100 KB.

- If you are using a disk array, create the smallest LUN the array will allow, or, on an HP Integrity server, you can partition a LUN; see "Creating a Disk Partition on an HP Integrity System".
- If you are using individual disks, use either a small disk, or a portion of a disk. On an HP Integrity server, you can partition a disk; see "Creating a Disk Partition on an HP Integrity System".

IMPORTANT: On HP 9000 systems, there is no means of partitioning a disk or LUN, so you will need to dedicate an entire small disk or LUN for the lock LUN. This means that in a mixed cluster containing both Integrity and HP-PA systems, you must also use an entire disk or LUN; if you partition the device as described below, the HP-PA nodes will not be able to see the partitions.

Creating a Disk Partition on an HP Integrity System

You can use the `idisk` utility to create a partition for a lock LUN in a cluster that will contain only HP Integrity servers. Use the procedure that follows; see the `idisk (1m)` manpage for more information. Do this on one of the nodes in the cluster that will use this lock LUN.

CAUTION: Before you start, make sure the disk or LUN that is to be partitioned has no data on it that you need. `idisk` will destroy any existing data.

1. Use a text editor to create a file that contains the partition information. You need to create at least three partitions, for example:

   ```
   3
   EFI 100MBHPUX 1MB
   HPUX 100%
   ```

   This defines:

   - A 100 MB EFI (Extensible Firmware Interface) partition (this is required)
   - A 1 MB partition that can be used for the lock LUN
   - A third partition that consumes the remainder of the disk is and can be used for whatever purpose you like.
2. Save the file; for example you might call it `partition.txt`.
3. Create the partition; for example (using `partition.txt` as input):

   `/usr/sbin/idisk -w -p -f partition.txt /dev/rdsk/c1t4d0`

   Or, on an HP-UX 11i v3 system using agile addressing (see "About Device File Names (Device Special Files)" (page 77)):

   `/usr/sbin/idisk -w -p -f partition.txt /dev/rdisk/disk12`

   NOTE: Device files for partitions cannot be cluster-wide DSFs (cDSFs). For more information about cDSFs, see "About Cluster-wide Device Special Files (cDSFs)" (page 99).

   This will create three device files, for example

   `/dev/dsk/c1t4d0s1`, `/dev/dsk/c1t4d0s2`, and `/dev/dsk/c1t4d0s3`

   or:

   `/dev/disk/disk12_p1`, /dev/disk/disk12_p2, and `/dev/disk/disk12_p3`

   NOTE: The first partition, identified by the device file `/dev/dsk/c1t4d0s1` or `/dev/disk/disk12_p1` in this example, is reserved by EFI and cannot be used for any other purpose.
4. Create the device files on the other cluster nodes.

   Use the command `insf -e` on each node. This will create device files corresponding to the three partitions, though the names themselves may differ from node to node depending on each node's I/O configuration.
5. Define the lock LUN; see "Defining the Lock LUN".

Defining the Lock LUN

Use `cmquerycl -L` to create a cluster configuration file that defines the lock LUN.

- If the pathname for the lock LUN is the same on all nodes, use a command such as:

  `cmquerycl -C $SGCONF/config.ascii -L /dev/dsk/c0t1d1 -n <node1> -n <node2>`

- If the pathname for the lock LUN is different on some nodes, you must specify the path on each node; for example (all on one line):

  `cmquerycl -C $SGCONF/config.ascii -n <node1> -L /dev/dsk/c0t1d1 -n <node2> -L /dev/dsk/c0t1d2`

These commands create a configuration file which you can apply to the cluster configuration when you are ready to do so; see "Distributing the Binary Configuration File " (page 189). See also "Specifying a Lock LUN" (page 180).

△ **CAUTION:** Once you have specified the lock LUN in the cluster configuration file, running `cmapplyconf (1m)` or `cmdeploycl (1m)` will destroy any data on the LUN.

Excluding Devices from Probing

When you run `cmquerycl (1m)` or `cmdeploycl (1m)` to configure the cluster, Serviceguard probes disk devices (devices with paths that begin with `/dev/dsk/` and `/dev/disk`) to see if they are configured under the LVM, LVM2, or VxVM volume manager. By design, devices such as CD and DVD drives are omitted, but occasionally such a device may use a description string that Serviceguard does not recognize. The description string appears in the `Description` field of the output of `ioscan -funC disk`, and you can see what types of device Serviceguard will exclude from probing by looking at the `TYPE` field of the file `/etc/cmcluster/cmignoretypes.conf`.

NOTE: You should not edit `cmignoretypes.conf`

If Serviceguard does not recognize a device, you will see an error such as the following when you run `cmquerycl`:

```
Error reading device /dev/dsk/c0t0d0 0x8
Note: Disks were discovered which are not in use by either LVM or VxVM.
      Use pvcreate(1M) to initialize a disk for LVM or,
      use vxdiskadm(1M) to initialize a disk for VxVM.
```

In this example, Serviceguard did not recognize the type of the device identified by `/dev/dsk/c0t0d0`. You can fix the problem by adding a line for this device to the file `/etc/cmcluster/cmnotdisk.conf`; in this example, you would add:

```
DEVICE_FILE="/dev/dsk/c0t0d0"
```

Now when you run `cmquerycl`, if you still have disks that are not being managed by LVM, LVM2 or VxVM, you will see output such as the following:

```
cmquerycl -C test.conf -n pig
Note: Disks were discovered which are not in use by either LVM or VxVM.
      Use pvcreate(1M) to initialize a disk for LVM or,
      use vxdiskadm(1M) to initialize a disk for VxVM.
```

IMPORTANT: The purpose of `cmnotdisk.conf` is to exclude specific devices, usually CD and DVD drives, that Serviceguard does not recognize and which should not be probed. Make sure you do not add a `DEVICE_FILE` entry in `cmnotdisk.conf` for any device that *should* be probed; that is, disk devices being managed by LVM or LVM2. Excluding any such device will cause `cmquerycl` to fail.

Setting Up and Running the Quorum Server

If you will be using a Quorum Server rather than a lock disk or LUN, the Quorum Server software must be installed on a system other than the nodes on which your cluster will be running, and must be running during cluster configuration.

For detailed discussion, recommendations, and instructions for installing, updating, configuring, and running the Quorum Server, see *HP Serviceguard Quorum Server Version A.04.00 Release Notes* at http://www.hp.com/go/hpux-serviceguard-docs -> HP Quorum Server Software.

Creating the Storage Infrastructure and file systems with LVM, VxVM and 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 several ways:

- for Logical Volume Manager, see "Creating a Storage Infrastructure with LVM" (page 168).

  Do this *before* you configure the cluster if you use a lock disk; otherwise it can be done before or after.

- for Veritas Volume Manager, see"Creating a Storage Infrastructure with VxVM" (page 174)

  Do this *before* you configure the cluster if you use a lock disk; otherwise it can be done before or after.

- for Veritas Cluster File System with CVM, see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190)

  Do this *after* you configure the cluster.

- for Veritas Cluster Volume Manager, see "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208)

  Do this *after* you configure the cluster.

You can also use a mixture of volume types, depending on your needs.

NOTE: 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. As of HP-UX 11i v3, the I/O subsystem performs load balancing and multipathing automatically.

Creating a Storage Infrastructure with LVM

This section describes how to configure disk storage using HP's Logical Volume Manager (LVM). It covers the following tasks:

NOTE: The procedures that follow describe the command-line method of configuring LVM storage. There are two other, more automated methods you can use.

- System Management Homepage

  You can use the System Management Homepage to create or extend volume groups and create logical volumes. From the System Management Homepage, choose `Disks and File Systems`. Make sure you create mirrored logical volumes with PVG-strict allocation; see"Using Mirrored Individual Data Disks" (page 169).

  When you have created the logical volumes and created or extended the volume groups, specify the file system that is to be mounted on the volume group, then proceed with"Distributing Volume Groups to Other Nodes" (page 172).

- cmpreparestg

  You can use `cmpreparestg (1m)` to accomplish the tasks described under "Creating Volume Groups" (page 169). See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you do not need to perform the procedures that follow, but it is a good idea to read them so that you understand what `cmpreparestg` does for you. Then proceed to "Making Physical Volume Group Files Consistent" (page 173).

If you have already done LVM configuration, skip ahead to "Configuring the Cluster " (page 177).

NOTE: The procedures that follow assume you are using LVM rather than LVM 2.0. For information about LVM 2.0 specifically, see the white paper *LVM 2.0 Volume Groups in HP-UX 11i v3*, which you can find under http://www.hp.com/go/hpux-core-docs.

For more information on using LVM in general, including LVM 2.0, see the Logical Volume Management volume of the HP-UX *System Administrator's Guide* at http://www.hp.com/go/hpux-core-docs. For more information about the tasks covered in this section, see Chapter 3 of that guide, and particularly the section on "Common LVM Tasks".

Using the EMS Disk Monitor

The Event Monitoring Service HA Disk Monitor allows you 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, see *Using High Availability Monitors* at the address given in the preface to this manual.

Using Mirrored Individual Data Disks

The procedures that follow use physical volume groups to allow mirroring of individual disks such that each logical volume is mirrored to a disk on a different I/O bus. This is known as PVG-strict mirroring.

Before you proceed, make sure your disk hardware is configured in such a way that a disk to be used as a mirror copy is connected to each node on a different bus from the bus that is used for the other (primary) copy.

Creating Volume Groups

NOTE: You can create volume groups by means of the `cmpreparestg (1m)` command. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you can skip this step and proceed to "Making Physical Volume Group Files Consistent" (page 173).

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/d*/*
```

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.

NOTE: Under agile addressing, the physical devices in these examples would have names such as `/dev/rdisk/disk1` and `/dev/rdisk/disk2`. See "About Device File Names (Device Special Files)" (page 77).

If you are using cDSFs, the device files would have names such as `/dev/rcdisk/disk1` and `/dev/rcdisk/disk2`. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

On the configuration node (`ftsys9`), use the `pvcreate(1m)` 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
```

Use the following procedure to build a volume group on the configuration node (`ftsys9`). Later, you will create the same volume group on other nodes; see "Distributing Volume Groups to Other Nodes" (page 172).

NOTE: If you are using the March 2008 version or later of HP-UX 11i v3, you can skip steps 1 and 2; `vgcreate (1m)` will create the device file for you.

1. Create the group directory; for example, `vgdatabase`:

   ```
   mkdir /dev/vgdatabase
   ```

2. 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 above. (This will avoid further reconfiguration later, when NFS-mounted logical volumes are created in the volume group.)

   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
   ```

△ **CAUTION:** Volume groups used by Serviceguard must have names no longer than 35 characters (that is, the name that follows `/dev/`, in this example `vgdatabase`, must be at most 35 characters long).

NOTE: If you are using cDSFs, you should be using them exclusively.

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.

4. Repeat steps 1–3 for additional volume groups.

Creating Logical Volumes

NOTE: You can create a single logical volume or multiple logical volumes by means of the `cmpreparestg (1m)` command. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) and the manpage for more information. If you use `cmpreparestg`, you can skip this step and proceed to "Making Physical Volume Group Files Consistent" (page 173).

Use a command such as the following to create a logical volume (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 copy of any given piece of data will be in a different physical volume group from the original.

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

Setting Logical Volume Timeouts

In the event that a I/O request to a logical volume never succeeds (for example, a vital set of disks fails), your application or file system may block indefinitely. To prevent this, you can set a timeout on the logical volume. If the device fails to respond within the timeout period, LVM will return an I/O error to the caller. Set the timeout value using the `-t` option of the `lvchange` command. This sets the timeout value in seconds for a logical volume. For example, to set the timeout for `/dev/vg01/lvol1` to one minute, enter the following command:

```
lvchange -t 60 /dev/vg01/lvol1
```

TIP: Set the logical volume timeout to an integral multiple of any timeout assigned to the underlying physical volumes. Otherwise, the actual duration of the I/O request can exceed the logical volume timeout. For details on how to change the I/O timeout value on a physical volume, see the manpage for `pvchange (1m)`.

Creating File Systems

If your installation uses file systems, create them next. If you want to use the Logical Volume group (LVM) as a shared LVM, then do not create a file system on the shared logical volume group.

NOTE: You can create file systems by means of the `cmpreparestg (1m)` command. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you can skip the procedure that follows, and proceed to "Making Physical Volume Group Files Consistent" (page 173).

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
   ```

Distributing Volume Groups to Other Nodes

After creating volume groups for cluster packages, 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.

NOTE: You can distribute the volume groups by means of the `cmpreparestg (1m)` command. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you can skip to "Making Physical Volume Group Files Consistent" (page 173).

Deactivating the Volume Group

NOTE: If you plan to use `cmpreparestg`, you can skip this step and proceed to "Making Physical Volume Group Files Consistent" (page 173).

At the time you create the volume group, it is active on the configuration node (`ftsys9`, for example). The next step is to unmount the file system and deactivate the volume group; for example, on `ftsys9`:

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

NOTE: Do this during this setup process only, so that activation and mounting can be done by the package control script at run time. You do not need to deactivate and unmount a volume simply in order to create a map file (as in step 1 of the procedure that follows).

Distributing the Volume Group

NOTE: If you use `cmpreparestg`, you can skip the procedure that follows and proceed to "Making Physical Volume Group Files Consistent" (page 173).

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 `ftsys10`, 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. Make sure the physical volume names are correct throughout the cluster.

When the volume group can be activated on this node, perform a vgcfgbackup. (This backup will be available 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 physical volume group. Note that on each cluster node, the /etc/lvmpvg file must contain the correct physical volume names for the physical volume groups's disks *as they are known on that node*. Physical volume names for the same disks could be different on different nodes. After distributing volume groups to other nodes, make sure each node's /etc/lvmpvg file correctly reflects the contents of all physical volume groups on that node. See "Making Physical Volume Group Files Consistent" (page 173).

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 this section if you do not use physical volume groups for mirrored individual disks in your disk configuration, or if you are using cDSFs; see "About Cluster-wide Device Special Files (cDSFs)" (page 99) for more information about cDSFs.

Different volume groups may be activated by different subsets of nodes within a Serviceguard cluster. In addition, if you are not using cDSFs, the physical volume name for any given disk may be different on one node from what 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.

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

1. Copy /etc/lvmpvg from ftsys9 to /etc/lvmvpg.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, copy all physical volume group entries for that volume group to/etc/lvmpvg.new.
4. Adjust any physical volume names in /etc/lvmvpg.new to reflect their correct names on ftsys10.
5. On ftsys10, copy /etc/lvmpvg to /etc/lvmpvg.old, to create a backup. Copy /etc/lvmpvg.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, which can vary from one node to another.

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, VEA, to carry out configuration tasks. For more information, see the Veritas Volume Manager documentation posted at http://www.hp.com/go/hpux-core-docs.

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 for your version of VxVM. 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. See "Migrating from LVM to VxVM Data Storage " (page 360) for more information 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`**

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

NOTE: You can use `cmpreparestg (1m)` to create a VxVM/CVM disk group. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you do not need to perform the procedures that follow, but it is a good idea to read them so that you understand what `cmpreparestg` does for you.

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

logdata       enabled             972078742.1084.node1
```

Creating Logical Volumes

NOTE: You can create a single logical volume or multiple logical volumes using `cmpreparestg (1m)`. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) and the manpage for more information. If you use `cmpreparestg`, you can skip this step, but it is a good idea to read them so that you understand what `cmpreparestg` does for you.

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.

NOTE: You can create file systems by means of the `cmpreparestg (1m)` command. See "Using Easy Deployment Commands to Configure the Cluster" (page 153) for more information. If you use `cmpreparestg`, you can skip the following steps, but it is a good idea to read them so that you understand what `cmpreparestg` does for you.

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 deported by the package control script on other 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 file systems 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 configuration file, nor in the package 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 using traditional Serviceguard commands. This must be done on a system that is not part of a Serviceguard cluster (that is, on which Serviceguard is installed but not configured).

NOTE: You can use Serviceguard Manager to configure a cluster: open the System Management Homepage (SMH) and choose `Tools-> Serviceguard Manager`. See "Using Serviceguard Manager" (page 23) for more information.

You can also use Easy Deployment commands; see"About Easy Deployment" (page 100) and "Using Easy Deployment Commands to Configure the Cluster" (page 153).

To use traditional Serviceguard commands to configure the cluster, follow directions in the remainder of this section.

Use the `cmquerycl (1m)` command to specify a set of nodes to be included in the cluster and to generate a template for the cluster configuration file.

IMPORTANT: See the entry for *`NODE_NAME`* under "Cluster Configuration Parameters " (page 105) for important information about restrictions on the node name.

Here is an example of the command (enter it all one line):

```
cmquerycl -v -C $SGCONF/clust1.conf -n ftsys9 -n ftsys10
```

This creates a template file, by default `/etc/cmcluster/clust1.conf`. In this output 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.

NOTE: HP strongly recommends that you modify the file so as to send the heartbeat over all possible networks. See also *`HEARTBEAT_IP`* under "Cluster Configuration Parameters " (page 105), and "Specifying the Address Family for the Heartbeat " (page 178).

The `cmquerycl(1m)` manpage further explains the calling parameters as well as those that appear in the template file. See also "Cluster Configuration Parameters " (page 105). Modify your `/etc/cmcluster/clust1.conf` file as needed.

cmquerycl Options

Speeding up the Process

In a larger or more complex cluster with many nodes, networks or disks, the `cmquerycl` command may take several minutes to complete. 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. This is the default when you use `cmquerycl` with the `-C` option.

(Do not use `-w local` if you need to discover nodes and subnets for a cross-subnet configuration; see "Full Network Probing" below.)

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

Specifying the Address Family for the Cluster Hostnames

You can use the `-a` option to tell Serviceguard to resolve cluster node names (as well as Quorum Server hostnames, if any) to IPv4 addresses only (`-a ipv4`) IPv6 addresses only (`-a ipv6`), or both (`-a any`). You can also configure the address family by means of the *`HOSTNAME_ADDRESS_FAMILY`* in the cluster configuration file.

IMPORTANT: See "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102) for a full discussion, including important restrictions for IPv6–only and mixed modes.

If you use the `-a` option, Serviceguard will ignore the value of the *`HOSTNAME_ADDRESS_FAMILY`* parameter in the existing cluster configuration, if any, and attempt to resolve the cluster and Quorum Server hostnames as specified by the `-a` option:

- If you specify `-a ipv4`, each of the hostnames must resolve to at least one IPv4 address; otherwise the command will fail.
- Similarly, if you specify `-a ipv6`, each of the hostnames must resolve to at least one IPv6 address; otherwise the command will fail.
- If you specify `-a any`, Serviceguard will attempt to resolve each hostname to an IPv4 address, then, if that fails, to an IPv6 address.

If you do not use the `-a` option:

- If a cluster is already configured, Serviceguard will use the value configured for *`HOSTNAME_ADDRESS_FAMILY`*, which defaults to IPv4.
- If no cluster configured, and Serviceguard finds at least one IPv4 address that corresponds to the local node's hostname (that is, the node on which you are running `cmquerycl`), Serviceguard will attempt to resolve all hostnames to IPv4 addresses. If no IPv4 address is found for a given hostname, Serviceguard will look for an IPv6 address. (This is the same behavior as if you had specified `-a any`.)

Specifying the Address Family for the Heartbeat

To tell Serviceguard to use only IPv4, or only IPv6, addresses for the heartbeat, use the `-h` option. For example, to use only IPv6 addresses:

```
cmquerycl -v -h ipv6 -C $SGCONF/clust1.conf -n ftsys9 -n ftsys10
```

- `-h ipv4` tells Serviceguard to discover and configure only IPv4 subnets. If it does not find any eligible subnets, the command will fail.
- `-h ipv6` tells Serviceguard to discover and configure only IPv6 subnets. If it does not find any eligible subnets, the command will fail.
- If you don't use the `-h` option, Serviceguard will choose the best available configuration to meet minimum requirements, preferring an IPv4 LAN over IPv6 where both are available. The resulting configuration could be IPv4 only, IPv6 only, or a mix of both. You can override Serviceguard's default choices by means of the *`HEARTBEAT_IP`* parameter, discussed under "Cluster Configuration Parameters " (page 105); that discussion also spells out the heartbeat requirements.
- The `-h` and `-c` options are mutually exclusive.

Specifying the Cluster Lock

You can use the `cmquerycl` command line to specify a cluster lock LUN (`-L lock_lun_device`), lock disk (`-L lock_vg: lock_pv`), or quorum server (`-q quorum_server [qs_ip2]`). See the `cmquerycl (1m)` manpage for details.

NOTE: You can specify only one lock disk on the command line; if you need to specify a second cluster lock disk, you must do so in the cluster configuration file.

For more information, see "Specifying a Lock Disk" (page 179), "Specifying a Lock LUN" (page 180), and "Specifying a Quorum Server" (page 180).

Generating a Network Template File

As of Serviceguard A.11.20, a separate form of `cmquerycl` discovers connected LAN interfaces on each node you specify, and writes it to a file you specify; for example:

```
cmquerycl -n node1 -n node2 -N mynetwork
```

You can now edit `mynetwork` to add IP address and subnet information for new interfaces which are not yet configured, and then use `cmapplyconf (1m)` to configure the new interfaces into the cluster; for example:

```
cmapplyconf -N mynetwork
```

IMPORTANT:

- You cannot use `cmapplyconf -N` if the cluster already exists; in that case, follow instructions under "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).
- You can only *add* information to the output file (`mynetwork` in this example); do not change the information already in the file.

For more information, see the `cmquerycl (1m)` and `cmapplyconf (1m)` manpages.

Full Network Probing

`-w full` lets you specify full network probing, in which actual connectivity is verified among all LAN interfaces on all nodes in the cluster, whether or not they are all on the same subnet.

NOTE: This option must be used to discover actual or potential nodes and subnets in a cross-subnet configuration. See "Obtaining Cross-Subnet Information" (page 181). It will also validate IP Monitor polling targets; see "Monitoring LAN Interfaces and Detecting Failure: IP Level" (page 70), and *`POLLING_TARGET`* under "Cluster Configuration Parameters " (page 105).

Specifying a Lock Disk

A cluster lock disk, lock LUN, or Quorum Server, is required for two-node clusters. The lock must be accessible to all nodes and must be powered separately from the nodes. See "Cluster Lock " (page 44)for more information.

To specify a lock disk, enter the lock disk information either on the command line, or in the configuration file. 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 template created with `cmquerycl` are the volume group and physical volume name of a disk connected to all cluster nodes; if there is more than one, the disk is chosen on the basis of 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.

For more information, see the discussion of these parameters under "Cluster Configuration Parameters " (page 105), the `cmquerycl (1m)` manpage, and the comments in the template configuration file.

To display the failover times of disks, use the `cmquerycl` command, specifying all the nodes in the cluster. The output of the command lists the disks connected to each node together with the re-formation time associated with each.

Do not include the node's entire domain name; for example, specify `ftsys9`, not `ftsys9.cup.hp.com`:

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

`cmquerycl` will not print out the re-formation time for a volume group that currently belongs to a cluster. If you want `cmquerycl` to print the re-formation time for a volume group, run `vgchange -c n <vg name>` to clear the cluster ID from the volume group. After you are done, do not forget to run `vgchange -c y vgname` to rewrite the cluster ID back to the volume group; for example:

```
vgchange -c y /dev/vglock
```

NOTE: You should not configure a second lock volume group or physical volume unless your configuration specifically requires it. See "Dual Lock Disk" (page 46).

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 *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; for example:

```
SECOND_CLUSTER_LOCK_VG /dev/vglock
SECOND_CLUSTER_LOCK_PV /dev/dsk/c4t0d0
```

or (using agile addressing; see "About Device File Names (Device Special Files)" (page 77)):

```
SECOND_CLUSTER_LOCK_VG /dev/vglock
SECOND_CLUSTER_LOCK_PV /dev/disk/disk100
```

NOTE: If you are using cDSFs, the device file would be in the `/dev/rdisk/` directory; for example `/dev/rdisk/disk100`. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

See also "Choosing Cluster Lock Disks" (page 164).

Specifying a Lock LUN

A cluster lock disk, lock LUN, or Quorum Server, is required for two-node clusters. The lock must be accessible to all nodes and must be powered separately from the nodes. See also "Cluster Lock " (page 44); and see "Setting Up a Lock LUN" (page 165) for configuration information and restrictions.

To specify a lock LUN in the cluster configuration file, enter the lock LUN information following each node name, for example:

```
NODE_NAME   hasupt21
  NETWORK_INTERFACE  lan1
    HEARTBEAT_IP  15.13.173.189
  NETWORK_INTERFACE  lan2
  NETWORK_INTERFACE  lan3
  CLUSTER_LOCK_LUN /dev/dsk/c0t1d1
```

Specifying a Quorum Server

IMPORTANT: The following are standard instructions. For special instructions that may apply to your version of Serviceguard and the Quorum Server see "Configuring Serviceguard to Use the Quorum Server" in the latest version of the *HP Serviceguard Quorum Server Version A.04.00 Release Notes*, at http://www.hp.com/go/hpux-serviceguard-docs under `HP Serviceguard Quorum Server Software`

A cluster lock LUN or Quorum Server is required for two-node clusters. To obtain a cluster configuration file that includes Quorum Server parameters, use the `-q` option of the `cmquerycl` command, specifying a Quorum Server hostname or IP address , for example (all on one line):

cmquerycl -q *<QS_Host>* -n ftsys9 -n ftsys10 -C *<ClusterName>*.conf

To specify an alternate hostname or IP address by which the Quorum Server can be reached, use a command such as (all on one line):

cmquerycl -q *<QS_Host>* *<QS_Addr>* -n ftsys9 -n ftsys10 -C *<ClusterName>*.conf

Enter the *QS_HOST* (IPv4 or IPv6), optional *QS_ADDR* (IPv4 or IPv6), *QS_POLLING_INTERVAL*, and optionally a *QS_TIMEOUT_EXTENSION*; and also check the *HOSTNAME_ADDRESS_FAMILY* setting, which defaults to IPv4. See the parameter descriptions under "Cluster Configuration Parameters " (page 105) for more information.

IMPORTANT: For important information, see also "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102); and "What Happens when You Change the Quorum Configuration Online" (page 47)

Obtaining Cross-Subnet Information

As of Serviceguard A.11.18 it is possible to configure multiple subnets, joined by a router, both for the cluster heartbeat and for data, with some nodes using one subnet and some another. See "Cross-Subnet Configurations" (page 29) for rules and definitions.

You must use the `-w full` option to `cmquerycl` to discover the available subnets.

For example, assume that you are planning to configure four nodes, `NodeA`, `NodeB`, `NodeC`, and `NodeD`, into a cluster that uses the subnets `15.13.164.0`, `15.13.172.0`, `15.13.165.0`, `15.13.182.0`, `15.244.65.0`, and `15.244.56.0`.

The following command

```
cmquerycl –w full –n nodeA –n nodeB –n nodeB –n nodeC –n nodeD
```

will produce the output such as the following:

```
Node Names:     nodeA
                nodeB
                nodeC
                nodeD

Bridged networks (full probing performed):

1       lan3            (nodeA)
        lan4            (nodeA)
        lan3            (nodeB)
        lan4            (nodeB)

2       lan1            (nodeA)
        lan1            (nodeB)

3       lan2            (nodeA)
        lan2            (nodeB)

4       lan3            (nodeC)
        lan4            (nodeC)
        lan3            (nodeD)
        lan4            (nodeD)
5       lan1            (nodeC)
        lan1            (nodeD)

6       lan2            (nodeC)
        lan2            (nodeD)

IP subnets:

IPv4:
```

```
15.13.164.0         lan1      (nodeA)
                    lan1      (nodeB)

15.13.172.0         lan1      (nodeC)
                    lan1      (nodeD)

15.13.165.0         lan2      (nodeA)
                    lan2      (nodeB)

15.13.182.0         lan2      (nodeC)
                    lan2      (nodeD)

15.244.65.0         lan3      (nodeA)
                    lan3      (nodeB)

15.244.56.0         lan4      (nodeC)
                    lan4      (nodeD)

IPv6:

3ffe:1111::/64      lan3      (nodeA)
                    lan3      (nodeB)

3ffe:2222::/64      lan3      (nodeC)
                    lan3      (nodeD)

Possible Heartbeat IPs:

15.13.164.0                       15.13.164.1         (nodeA)
                                  15.13.164.2         (nodeB)

15.13.172.0                       15.13.172.158       (nodeC)
                                  15.13.172.159       (nodeD)

15.13.165.0                       15.13.165.1         (nodeA)
                                  15.13.165.2         (nodeB)

15.13.182.0                       15.13.182.158       (nodeC)
                                  15.13.182.159       (nodeD)

Route connectivity(full probing performed):

1       15.13.164.0
        15.13.172.0

2       15.13.165.0
        15.13.182.0

3       15.244.65.0

4       15.244.56.0
```

In the `Route connectivity` section, the numbers on the left (`1-4`) identify which subnets are routed to each other (for example `15.13.164.0` and `15.13.172.0`).

IMPORTANT: Note that in this example subnet `15.244.65.0`, used by `NodeA` and `NodeB`, is not routed to `15.244.56.0`, used by `NodeC` and `NodeD`.

But subnets `15.13.164.0` and `15.13.165.0`, used by `NodeA` and `NodeB`, *are* routed respectively to subnets `15.13.172.0` and`15.13.182.0`, used by `NodeC` and `NodeD`. At least one such routing among all the nodes must exist for `cmquerycl` to succeed.

For information about configuring the heartbeat in a cross-subnet configuration, see the *`HEARTBEAT_IP`* parameter discussion under "Cluster Configuration Parameters " (page 105).

Identifying Heartbeat Subnets

The cluster configuration file includes entries for IP addresses on the heartbeat subnet. HP recommends that you use a dedicated heartbeat subnet, and configure heartbeat on other subnets as well, including the data subnet. The heartbeat can be configured on an IPv4 or IPv6 subnet; see "About Hostname Address Families: IPv4-Only, IPv6-Only, and Mixed Mode" (page 102).

The heartbeat can comprise multiple subnets joined by a router. In this case at least two heartbeat paths must be configured for each cluster node. See also the discussion of *`HEARTBEAT_IP`* under "Cluster Configuration Parameters " (page 105), and "Cross-Subnet Configurations" (page 29).

Specifying Maximum Number of Configured Packages

This specifies the most packages that can be configured in the cluster.

The parameter value must be equal to or greater than the number of packages currently configured in the cluster. The count includes all types of packages: failover, multi-node, and system multi-node.

The default is 300, which is the maximum allowable number of packages in a cluster.

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 the MEMBER_TIMEOUT Parameter

The `cmquerycl` command supplies a default value of 14 seconds for the *`MEMBER_TIMEOUT`* parameter. Changing this value will directly affect the cluster's re-formation and failover times. You may need to increase the value if you are experiencing cluster node failures as a result of heavy system load or heavy network traffic; or you may need to decrease it if cluster re-formations are taking a long time.

You can change *`MEMBER_TIMEOUT`* while the cluster is running.

For more information, see the *`MEMBER_TIMEOUT`* parameter discussion under "Cluster Configuration Parameters " (page 105), "What Happens when a Node Times Out" (page 85), and "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low" (page 320).

Controlling Access to the Cluster

Serviceguard access-control policies define cluster users' administrative or monitoring capabilities.

A Note about Terminology

Although you will also sometimes see the term role-based access (RBA) in the output of Serviceguard commands, the preferred set of terms, always used in this manual, is as follows:

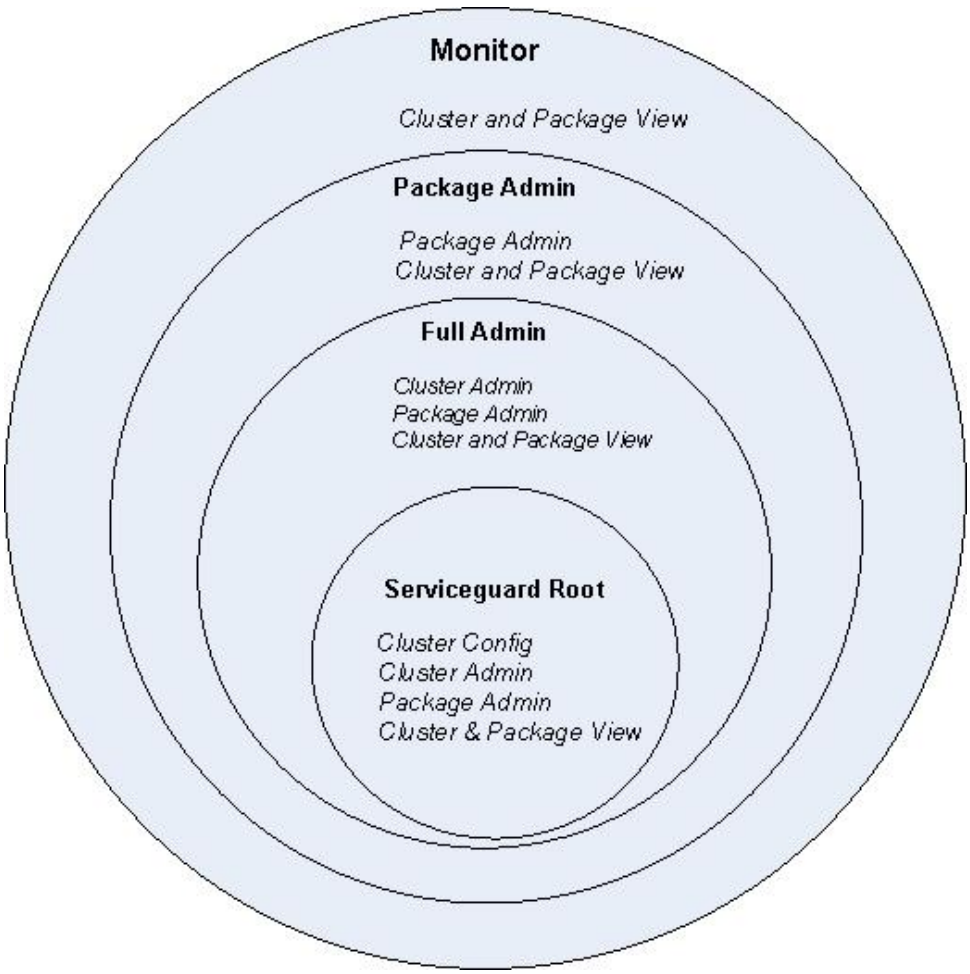
- Access-control policies- the set of rules defining user access to the cluster.
  - Access-control policy - one of these rules, comprising the three parameters *USER_NAME*, *USER_HOST*, *USER_ROLE*. See "Setting up Access-Control Policies" (page 185).
- Access roles - the set of roles that can be defined for cluster users (Monitor, Package Admin, Full Admin).
  - Access role - one of these roles (for example, Monitor).

How Access Roles Work

Serviceguard daemons grant access to Serviceguard commands by matching the command user's hostname and username against the access control policies you define. Each user can execute only the commands allowed by his or her role.

The diagram that shows the access roles and their capabilities. The innermost circle is the most trusted; the outermost the least. Each role can perform its own functions and the functions in all of the circles outside it. For example Serviceguard Root can perform its own functions plus all the functions of Full Admin, Package Admin and Monitor; Full Admin can perform its own functions plus the functions of Package Admin and Monitor; and so on.

Figure 36 Access Roles



Levels of Access

Serviceguard recognizes two levels of access, root and non-root:

- Root access: *Full capabilities; only role allowed to configure the cluster.*

  As Figure 36 shows, users with root access have complete control over the configuration of the cluster and its packages. This is the only role allowed to use the `cmcheckconf`, `cmapplyconf`, `cmdeleteconf`, and `cmmodnet -a` commands.

  In order to exercise this Serviceguard role, you must log in as the HP-UX root user (superuser) on a node in the cluster you want to administer. Conversely, the HP-UX root user on any node in the cluster always has full Serviceguard root access privileges for that cluster; no additional Serviceguard configuration is needed to grant these privileges.

IMPORTANT: Users on systems outside the cluster can gain Serviceguard root access privileges to configure the cluster only via a secure connection (`rsh` or `ssh`).

- Non-root access: Other users can be assigned one of four roles:
  - Full Admin: *Allowed to perform cluster administration, package administration, and cluster and package view operations*.

    These users can administer the cluster, but cannot configure or create a cluster. Full Admin includes the privileges of the Package Admin role.
  - (all-packages) Package Admin: *Allowed to perform package administration, and use cluster and package view commands.*

    These users can run and halt any package in the cluster, and change its switching behavior, but cannot configure or create packages. Unlike single-package Package Admin, this role is defined in the cluster configuration file. Package Admin includes the cluster-wide privileges of the Monitor role.
  - (single-package) Package Admin: *Allowed to perform package administration for a specified package, and use cluster and package view commands.*

    These users can run and halt a specified package, and change its switching behavior, but cannot configure or create packages. This is the only access role defined in the package configuration file; the others are defined in the cluster configuration file. Single-package Package Admin also includes the cluster-wide privileges of the Monitor role.
  - Monitor: *Allowed to perform cluster and package view operations.*

    These users have read-only access to the cluster and its packages.

IMPORTANT: A remote user (one who is not logged in to a node in the cluster, and is not connecting via `rsh` or `ssh`) can have only Monitor access to the cluster.

(Full Admin and Package Admin can be configured for such a user, but this usage is deprecated and in a future release may cause `cmapplyconf` and `cmcheckconf` to fail. As of Serviceguard A.11.18 configuring Full Admin or Package Admin for remote users gives them Monitor capabilities. See "Setting up Access-Control Policies" (page 185) for more information.)

Setting up Access-Control Policies

The HP-UX root user on each cluster node is automatically granted the Serviceguard root access role on all nodes. (See "Configuring Root-Level Access" (page 157) for more information.) Access-control policies define non-root roles for other cluster users.

NOTE: For more information and advice, see the white paper *Securing Serviceguard* at http://www.hp.com/go/hpux-serviceguard-docs.

Define access-control policies for a cluster in the cluster configuration file; see "Cluster Configuration Parameters " (page 105). You can define up to 200 access policies for each cluster. A root user can create or modify access control policies while the cluster is running.

Define policies for a specific package in the package configuration file; see the entries for *user_name* and related package-configuration parameters (page 240).

NOTE: Once nodes are configured into a cluster, the access-control policies you set in the cluster and package configuration files govern cluster-wide security; changes to the "bootstrap" `cmclnodelist` file are ignored (see "Allowing Root Access to an Unconfigured Node" (page 157)).

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

- Each *USER_NAME* can consist either of the literal `ANY_USER`, or a maximum of 8 login names from the `/etc/passwd` file on *USER_HOST*. The names must be separated by spaces or tabs, for example:

  ```
  # Policy 1:
  USER_NAME john fred patrick
  USER_HOST bit
  USER_ROLE PACKAGE_ADMIN
  ```

- *USER_HOST* is the node where *USER_NAME* will issue Serviceguard commands.

  NOTE: The commands must be issued on*USER_HOST* but can take effect on other nodes; for example `patrick` can use `bit`'s command line to start a package on `gryf`.

  Choose one of these three values for *USER_HOST*:

  - `ANY_SERVICEGUARD_NODE` - any node on which Serviceguard is configured, and which is on a subnet with which nodes in this cluster can communicate (as reported by`cmquerycl -w full`).

    NOTE: If you set *USER_HOST* to `ANY_SERVICEGUARD_NODE`, set *USER_ROLE* to `MONITOR`; users connecting from outside the cluster cannot have any higher privileges (unless they are connecting via `rsh` or `ssh`; this is treated as a local connection).

    Depending on your network configuration, `ANY_SERVICEGUARD_NODE` can provide wide-ranging read-only access to the cluster.

  - `CLUSTER_MEMBER_NODE` - any node in the cluster
  - A specific node name - Use the hostname portion (the first of four parts) of a fully-qualified domain name that can be resolved by the name service you are using; it should also be in each node's `/etc/hosts`. Do not use an IP addresses or the fully-qualified domain name. If there are multiple hostnames (aliases) for an IP address, one of those must match *USER_HOST*. See "Configuring Name Resolution" (page 159) for more information.

- *USER_ROLE* must be one of these three values:

  - `MONITOR`
  - `FULL_ADMIN`
  - `PACKAGE_ADMIN`

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

only. These roles are not exclusive; for example, you can configure more than one PACKAGE_ADMIN for the same package.

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

Here is an example of an access control policy:

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

If this policy is defined in the cluster configuration file, it grants user john the PACKAGE_ADMIN role for any package on node bit. User john also has the MONITOR role for the entire cluster, because PACKAGE_ADMIN includes MONITOR. If the policy is defined in the package configuration file for PackageA, then user john on node bit has the PACKAGE_ADMIN role only for PackageA.

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

Role Conflicts

Do not configure different roles for the same user and host; Serviceguard treats this as a conflict and will fail with an error when applying the configuration. "Wildcards", such as ANY_USER and ANY_SERVICEGUARD_NODE, are an exception: it is acceptable for ANY_USER and john to be given different roles.

IMPORTANT: Wildcards do not degrade higher-level roles that have been granted to individual members of the class specified by the wildcard. For example, you might set up the following policy to allow root users on remote systems access to the cluster:

```
USER_NAME root
USER_HOST ANY_SERVICEGUARD_NODE
USER_ROLE MONITOR
```

This *does not* reduce the access level of users who are logged in as root on nodes in this cluster; they will always have full Serviceguard root-access capabilities.

Consider what would happen if these entries were in the cluster configuration file:

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

# Policy 2:
USER_NAME john
USER_HOST bit
USER_ROLE MONITOR

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

In the above example, the configuration would fail because user john is assigned two roles. (In any case, Policy 2 is unnecessary, because PACKAGE_ADMIN includes the role of MONITOR.)

Policy 3 does not conflict with any other policies, even though the wildcard ANY_USER includes the individual user john.

NOTE: Check spelling especially carefully when typing wildcards, such as `ANY_USER` and `ANY_SERVICEGUARD_NODE`. If they are misspelled, Serviceguard will assume they are specific users or nodes.

Package versus Cluster Roles

Package configuration will fail if there is any conflict in roles between the package configuration and the cluster configuration, so it is a good idea to have the cluster configuration file in front of you when you create roles for a package; use `cmgetconf` to get a listing of the cluster configuration file.

If a role is configured for a username/hostname in the cluster configuration file, do not specify a role for the same username/hostname in the package configuration file; and note that there is no point in assigning a package administration role to a user who is root on any node in the cluster; this user already has complete control over the administration of the cluster and its packages.

Adding Volume Groups

Add any LVM volume groups you have configured to the cluster configuration file, with a separate *`VOLUME_GROUP`* entry 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 that activates a volume group; see *`vg`* (page 236).

NOTE: If you are using CVM disk groups, they should be configured after cluster configuration is done, using the procedures described in "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208). Add CVM disk groups to the package configuration file; see *`cvm_dg`* (page 236).

Verifying the Cluster Configuration

If you have edited a 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.conf
```

The following items are checked:

- Network addresses and connections, and that all IP addresses configured into the cluster are in each node's `/etc/hosts` file.
- Cluster lock connectivity (if you are configuring a lock disk).
- Validity of configuration parameters for the cluster and packages.
- Uniqueness of names.
- Existence and permissions of scripts specified on the command line.
- That all nodes specified are in the same heartbeat subnet, except in cross-subnet configurations(page 29) .
- That the configuration filename is correcy.
- That all nodes are reachable.
- That no more than one *`CLUSTER_NAME`* and *`AUTO_START_TIMEOUT`* are specified.
- That the value for package run and halt script timeouts is less than 4294 seconds.
- That the value for *`AUTO_START_TIMEOUT`* variables is >=0.
- That the heartbeat network minimum requirement is met. See the entry for *`HEARTBEAT_IP`* under "Cluster Configuration Parameters " (page 105) .
- That at least one *`NODE_NAME`* is specified.

- That each node is connected to each heartbeat network.
- That all heartbeat networks are of the same type of LAN.
- That network interface device files specified are valid LAN device files.
- That *VOLUME_GROUP* entries are marked as cluster-aware.

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.

For more information, see the manpage for `cmcheckconf (1m)` and "Checking Cluster Components" (page 261).

Distributing the Binary Configuration File

After specifying all cluster parameters, apply the configuration. This action distributes the binary configuration file to all the nodes in the cluster. HP recommends doing this separately before you configure packages (as described in the next chapter) so 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.

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.conf
  ```

  or

  ```
  cmapplyconf -k -v -C /etc/cmcluster/clust1.ascii
  ```

  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.

NOTE: The apply will not complete unless the cluster lock volume group is activated on *exactly one* node before applying. There is one exception to this rule: a cluster lock had been previously configured on the same physical volume and volume group.

After the configuration is applied, the cluster lock volume group must be deactivated.

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, especially 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 the System Management Homepage (SMH), SAM, or 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 Veritas Cluster File System (CFS)

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CFS (and CVM - Cluster Volume Manager) at http://www.hp.com/go/hpux-serviceguard-docs.

This section provides information about configuring a cluster that uses the Veritas cluster file system (CFS) with Veritas cluster volume manager (CVM) 4.1 and later. Serviceguard enables you to create and manage disk groups and mount points using modular style packages or legacy style packages using CFS; however, creation and administration of modular CFS packages is supported with CVM 5.0.1 HP Serviceguard A.11.20 April 2011 patch and later. Both legacy CFS packages and modular CFS packages can co-exist in the same cluster. Configure CVM/CFS after you configure the cluster. Do not configure applications as a part of CVM disk groups or CFS mount point packages. The Veritas volume manager includes a Java-based GUI, known as the Veritas Enterprise Administrator (VEA). When you create CVM disk groups and mount the volume using VEA, a legacy CFS package is created by default. Modular CFS packages cannot be created and managed using VEA; Serviceguard Manager can be used to create modular CFS packages, but not legacy CFS packages. For more information, see the Serviceguard Manager online help.

IMPORTANT: Serviceguard Manager must be used to create modular CFS packages. VEA cannot be used to create modular CFS packages. VEA can only be used to manage and administer legacy CFS packages.

Modular CFS packages v/s Legacy CFS packages

The following list helps you understand the difference between modular CFS packages and legacy CFS packages:

Table 8 Differences between Legacy CFS and Modular CFS

Legacy	Modular
The CFS legacy disk groups and mount points can be created or managed only by using the `cfsdgadm`, `cfsmntadm`, `cfsmount`, or `cfsumount`.	Unlike the legacy CFS packages, the modular CVM disk groups and CFS mount points cannot be created or managed by using the `cfs` commands. These are created and managed using `cmmakepkg`, `cmcheckconf`, `cmapplyconf`, `cmdeleteconf`, `cmrunpkg`, `cmhaltpkg`, or `cmmodpkg` commands.
Each package can only correspond to one disk group or one mount point leaving fewer packages for other purposes, so if you have many packages they may take a longer time for startup, shutdown, cmviewcl, etc.	Multiple disk groups and mount points can be consolidated into a single package. This significantly reduces the number of packages used, leaving more packages for other applications, thus improving the overall system performance for startup, shutdown, cmviewcl, etc.
You must not manually edit the packages, but use the `cfs` commands to edit the package. The disadvantage is that if you have multiple packages, you must edit each of them using the commands.	You must not edit the package using the `cfs` commands, but edit the parameters in the package configuration file if you want to manage the package. Also, if you have multiple disk groups and mount points in a package, you must halt the entire package if you want to modify one of them.
When a large number of packages are configured , the cluster information displayed by the `cmviewcl` command contains too many entries.	The high availability cluster information displayed by the `cmviewcl` command output is more compact.

Listed below are some of the operational differences in terms of commands used for creating or managing legacy and modular CFS packages. For usage, syntax, and keyword descriptions, see the respective Serviceguard man page of each command.

Table 9 Operational commands for Legacy CFS and Modular CFS

Operation	Commands used by legacy style	Equivalent commands in modular style
Create and run a disk group package	cfsdgadm add cfsdgadm activate	• cmmakepkg –m sg/cfs_all <package_ascii_file> Edit the package configuration file for the disk group parameters • cmapplyconf -P <package_ascii_file> • cmrunpkg <package name> **NOTE:** For adding another disk group to an existing modular CFS package, edit the configuration file to add the disk group parameters and run cmapplyconf. This operation can be performed only in the offline mode, i.e., when the package is not running.
Halt and delete a disk group package	cfsdgadm deactivate cfsdgadm delete	If it is the last disk group in the package: cmhaltpkg cmdeleteconf If it is not the last disk group in the package: • cmhaltpkg • Remove the corresponding cvm_disk_group entry from the configuration file • cmapplyconf • cmrunpkg
Displaying the disk group package attributes	cfsdgadm show_package cfsdgadm show_autorun cfsdgadm display	cmviewcl —v —p <package_name>

Table 9 Operational commands for Legacy CFS and Modular CFS *(continued)*

Operation	Commands used by legacy style	Equivalent commands in modular style
Modifying attributes for a disk group in a package	cfsdgadm modify	For modifying the attributes in an offline mode: • cmhaltpkg • edit parameters in the configuration file • cmapplyconf with modified parameters • cmrunpkg For modifying the attributes in an online mode: • edit parameters in the configuration file • cmapplyconf with modified parameters
Create and run a mount point package	cfsmntadm add cfsmount	Same as disk group, but edit the package configuration file for mount point parameters.
Create and run a check point package	cfsmntadm add ckpt cfsmount	Same as disk group, but edit the package configuration file for check point parameters.
Create and run a snapshot package	cfsmntadm add snapshot cfsmount	Same as disk group, but edit the package configuration file for snapshot parameters.
Halt and delete a mount point package	cfsumount cfsmntadm delete	If it is the last mount point in the package: cmhaltpkg cmdeleteconf If it is not the last mount point in the package: • cmhaltpkg • Remove the corresponding cfs_mount_point entry from the configuration file • cmapplyconf • cmrunpkg
Displaying the mount point package attributes	cfsmntadm show_package cfsmntadm show_autorun cfsmntadm display	cmviewcl —v —p <package_name>
Modifying attributes for a mount point in a package	cfsmntadm modify	For modifying the attributes in an offline mode: • cmhaltpkg • edit parameters in the configuration file • cmapplyconf with modified parameters • cmrunpkg For modifying the attributes in an online mode: • edit parameters in the configuration file • cmapplyconf with modified parameters

For information on Managing Disk Groups and Mount Points, see:

- "Managing Disk Groups and Mount Points Using Modular Packages" (page 195)
- "Managing Disk Groups and Mount Points Using Legacy Packages" (page 205)

NOTE: The "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208) section explains how to configure Veritas Cluster Volume Manager (CVM) disk groups *without* CFS; that is, for raw access only. Both solutions use many of the same commands, but in a slightly different order.

Refer to the Serviceguard man pages for more information about the commands `cfscluster`, `cfsdgadm`, `cfsmntadm`, `cfsmount`, `cfsumount`, and `cmgetpkgenv`. Information is also in the documentation for HP Serviceguard Storage Management Suite posted at http://www.hp.com/go/hpux-serviceguard-docs.

IMPORTANT: Before you proceed, make sure you have read "Planning Veritas Cluster Volume Manager (CVM) and Cluster File System (CFS)" (page 121), which contains important information and cautions.

Preparing the Cluster and the System Multi-node Package

The Veritas cluster volumes are managed by a Serviceguard-supplied system multi-node package, `SG-CFS-pkg`, which runs on all nodes at once, and cannot fail over.

The package for CVM 4.1 and later has the following responsibilities:

- Maintain Veritas configuration files `/etc/llttab`, `/etc/llthosts`, `/etc/gabtab`
- Launch required services: `cmvxd`, `cmvxpingd`, `vxfsckd`
- Start/halt Veritas processes in the proper order: `llt`, `gab`, `vxfen`, `odm`, `cvm`, `cfs`

NOTE: Do not edit the configuration file `SG-CFS-pkg.conf`. Create and modify configuration using the `cfs` administration commands.

1. First, make sure the cluster is running:

   `cmviewcl`

2. If it is not, start it:

   `cmruncl`

3. If you have not initialized your disk groups, or if you have an old install that needs to be re-initialized, use the `vxinstall` command to initialize VxVM/CVM disk groups. See "Initializing the Veritas Volume Manager " (page 209).
4. Activate the `SG-CFS-pkg` and start up CVM with the `cfscluster` command; this creates `SG-CFS-pkg`, and also starts it.

   This example, for the cluster file system, uses a timeout of 900 seconds; if your CFS cluster has many disk groups and/or disk LUNs visible to the cluster nodes, you may need to a longer timeout value. Use the `-s` option to start the CVM package in shared mode:

   `cfscluster config -t 900 -s`

5. Verify the system multi-node package is running and CVM is up, using the `cmviewcl` or `cfscluster` command. Following is an example of using the `cfscluster` command. In the last line, you can see that CVM is up, and that the mount point is not yet configured:

   `cfscluster status`

```
Node             :  ftsys9
Cluster Manager  :  up
CVM state        :  up (MASTER)
MOUNT POINT   TYPE    SHARED VOLUME   DISK GROUP    STATUS

Node             :  ftsys10
Cluster Manager  :  up
CVM state        :  up
MOUNT POINT   TYPE    SHARED VOLUME   DISK GROUP    STATUS
```

NOTE: Because the CVM system multi-node package automatically starts up the Veritas processes, *do not* edit `/etc/llthosts`, `/etc/llttab`, or `/etc/gabtab`.

The `cfscluster status` command displays the status of the disk groups and mount point packages created only for legacy CFS packages.

Creating the Disk Groups

Initialize the disk group from the master node.

1. Find the master node using `vxdctl` or `cfscluster status`
2. Initialize a new disk group, or import an existing disk group, in shared mode, using the `vxdg` command.
   - For a *new* disk group use the `init` option:

     `vxdg -s init logdata c4t0d6`
   - For an *existing* disk group, use the `import` option:

     `vxdg -C -s import logdata`
3. Verify the disk group. The state should be enabled and shared:

   `vxdg list`

```
NAME        STATE                    ID
logdata  enabled, shared, cds      11192287592.39.ftsys9
```

Creating the Disk Group Cluster Packages

1. Use the `cfsdgadm` command to create the package `SG-CFS-DG-`*`ID#`*, where *`ID#`* is an automatically incremented number, assigned by Serviceguard when it creates the package. In this example, the `SG-CFS-DG-`*`ID#`* package will be generated to control the disk group `logdata`, in shared write mode:

   `cfsdgadm add logdata all=sw`
2. You can verify the package creation with the `cmviewcl` command, or with the `cfsdgadm display` command. An example of `cfsdgadm` output is shown below:

   `cfsdgadm display`

```
 Node Name : ftsys9 (MASTER)
 DISK GROUP         ACTIVATION MODE
  logdata            off     (sw)

 Node Name :  ftsys10
  DISK GROUP        ACTIVATION MODE
   logdata           off     (sw)
```

3. Activate the disk group and start up the package:

   `cfsdgadm activate logdata`
4. To verify, you can use `cfsdgadm` or `cmviewcl`. This example shows the `cfsdgadm` output:

   `cfsdgadm display -v logdata`

```
NODE NAME        ACTIVATION MODE
  ftsys9         sw (sw)
     MOUNT POINT          SHARED VOLUME      TYPE
  ftsys10        sw (sw)
     MOUNT POINT         SHARED VOLUME       TYPE
```

5. To view the name of the package that is monitoring a disk group, use the `cfsdgadm show_package` command:

   `cfsdgadm show_package logdata`

```
SG-CFS-DG-1
```

Creating Volumes

1. Make `log_files` volume on the `logdata` disk group:

   `vxassist -g logdata make log_files 1024m`

2. Use the `vxprint` command to verify:

   `vxprint log_files`

```
disk group: logdata
TY NAME      ASSOC      KSTATE     LENGTH    PLOFFS  STATE  TUTIL0   PUTIL0
v  log_files    fsgen   ENABLED    1048576     -    ACTIVE   -        -
pl log_files-01 fsgen   ENABLED    1048576     -    ACTIVE   -        -
sd ct4t0d6-01   fsgen   ENABLED    1048576     -    ACTIVE   -        -
```

Managing Disk Groups and Mount Points Using Modular Packages

In a Serviceguard cluster, many disk group and mount point packages are created for applications using CVM and CFS and the applications are dependent on these packages. These disk group and mount point packages are configured for activating disk groups and mounting the volumes. For an application in a cluster, Serviceguard provides the flexibility to combine the CVM disk groups and CFS mount points into a single modular package. This feature is supported from Veritas CVM/CFS version 5.0.1 with HP Serviceguard A.11.20 April 2011 patches and later. As a result, the number of packages used for configuring disk groups and mount points in that cluster reduces significantly in comparison to the packages used for configuring CVM disk groups and CFS mount points in the legacy style of packaging. For example, an application requiring 100 disk groups, 100 mount points, 10 check points, and 10 snapshot packages would translate to a total of 220 packages in the legacy style of packaging. However, in the modular style of packaging, the 100 disk groups and 100 mount point packages can be consolidated into one single package, 10 check points into one package, and 10 snap shots into one package, reducing the total number of packages to three, thereby aiding manageability of packages in that cluster. Also, the number of packages available for configuring applications in the cluster increases. The high availability cluster information displayed by the `cmviewcl` command is more compact. Note that to create or modify modular CFS packages, CVM should be up and running.

CAUTION: While Serviceguard supports up to 300 configured packages, it is advised to maintain a manageable level when consolidating the disk groups and mount points in the package. The more disk groups and mount points in a package, the longer it takes that package to start up and shutdown, which may have an effect on any other packages that have dependencies on this package.

NOTE: Serviceguard supports online addition/removal of nodes from modular CFS packages. Specifying services and resources are not recommended for modular CFS packages. You cannot create or manage modular CFS packages for disk groups and mount points using VEA GUI; you can use Serviceguard Manager to do it.

External scripts and External Pre-scripts can be configured as a part of Modular CFS packages. It is recommended to keep the execution time of these scripts as short as possible. If the script execution time is huge then it might take a long time to start up the package resulting in bringing down the performance. See "About External Scripts" (page 142) for more information on external scripts.

Just like the legacy packages, the modular packages are dependent on system multi-node package, `SG-CFS-pkg`. For more information, see "Preparing the Cluster and the System Multi-node Package" (page 193) .

You cannot manage modular CFS packages using the legacy CFS commands `cfsdgadm`, `cfsmntadm`, `cfsmount`, and `cfsumount`. You must use the `cmmakepkg`, `cmapplyconf`, `cmrunpkg`, `cmhaltpkg`, and `cmdeleteconf` commands. The `cmmakepkg` command enables

you to create a package configuration file. The command uses the `sg/cfs` module that contains attributes for CVM disk groups, mount points, storage snapshot, and checkpoint parameters. A modular CFS package template can be generated using the `sg/cfs_all` template. The modular CFS packages run as a multi-node package and have a dependency on CVM system multi-node package, `SG-CFS-pkg`. The dependency on `SG-CFS-pkg` is automatically configured in the modular CFS package configuration file created using the `sg/cfs_all` template. CVM must be up and running when you create or modify modular CFS packages.

IMPORTANT: It is strongly recommended not to remove the dependency on `SG-CFS-pkg` that is automatically configured in the package configuration file.

The `cmrunpkg` and `cmhaltpkg` commands allow you to run and halt the modular CFS packages respectively. The `SG-CFS-pkg` package must be running, before the modular CFS packages can be started.

Refer to the Serviceguard man pages for more information about the commands `cmmakepkg (1m)`, `cmapplyconf (1m)`, `cfsdgadm (1m)`, `cfsmntadm (1m)`, `cmrunpkg (1m)`, `cmhaltpkg (1m)`, and `cmdeleteconf (1m)`.

Serviceguard enables you to consolidate resources required for an application as follows:

- You can consolidate all disk groups and mount points for an application into a single modular package as depicted in "Creating Modular Disk Group and Mount Point Packages" (page 196). You can also configure disk groups and mount points in separate packages. If you create the packages separately, you must configure the dependency of the mount point package on the disk group package explicitly in the package configuration file.
- You can consolidate all storage checkpoint packages and convert them to a single modular package.
- You can consolidate all snapshot packages and convert them to a single modular package.
- You can consolidate disk groups and snapshots into a single modular package.

Note that the dependency on the mount point package is a must for both checkpoint and snapshot packages.

It is recommended to configure all CVM disk group and CFS mount points using one style of packaging, legacy or modular. It is also recommended to use modular style packaging to reduce the number of multi-node packages consumed by CFS resources. During migration to modular style, both legacy and modular style CFS packages of different applications might coexist but HP strongly recommends completing the migration of all legacy CFS packages belonging to an application to modular style for consistency of administration. For guidelines on migrating from legacy CFS packages to modular CFS packages, see "Guidelines for Migrating from Legacy CFS Package to Modular CFS Package" (page 202).

For instructions on creating modular CFS packages, see:

- "Creating Modular Disk Group and Mount Point Packages" (page 196).
- "Creating Modular Checkpoint and Snapshot Packages for CFS" (page 199).

Creating Modular Disk Group and Mount Point Packages

1. Create a package configuration file:

   ```
   cmmakepkg -m sg/cfs_all /etc/cmcluster/cfspkg1.ascii
   Package template is created.
   This file must be edited before it can be used.
   ```

2. Edit the following package parameters in the `cfspkg1.ascii` package configuration file:

   ```
   package_name            cfspkg1

   node_name               node1
   node_name               node2
   ```

```
node_name               node3
node_name               node4

cvm_disk_group          cvm_dg1_node1
cvm_activation_mode     "node1=sw node2=sw node3=sw node4=sw"

cfs_mount_point         /mnt2
cfs_volume              cvm_dg1_node1/lvol1
cfs_mount_options       "node1=cluster node2=cluster node3=cluster node4=cluster"
cfs_primary_policy      ""
```

NOTE: By default, the package parameters are commented. You must uncomment them and specify the values.

If you wish to run the disk group and mount point packages on all nodes that the package can run, you must specify "all=cluster" as the value for the cfs_mount_options and "all=sw" for cvm_activation_mode in the `cfspkg1.ascii` file. Note that "all=cluster" does not configure the packages to run on all nodes in the cluster, but only on the nodes configured in the package.

The dependency parameters must be specified only if the disk groups and mount points are in separate packages.

cvm_disk_group	name of the shared disk group
cvm_activation_mode	activation mode for the shared disk group. Legal values are "sw", "sr", "ew", and "ro", which expands to shared write, shared read, exclusive write, and read only.
cfs_mount_point	unique mount point name; supports up to two levels of nested mount points
cfs_volume	Diskgroups/Logical volume pair used by the CFS mount point.
cfs_mount_options	the CFS mount options for the mount point. Legal values are "cluster", "tmplog", "delaylog", etc. For detailed information about the mount options, see the `mount_vxfs (1m)` manpage.
cfs_primary_policy	sets the primary migration policy for the mount point. This option sets up the order in which hosts assume the cluster file system primary ownership if the primary node fails. If no host specified in the list is active when the primary node fails, a node is chosen at random to be the new primary. When the policy is set, if the first host name specified in the list is not the current primary, a primary migration is triggered to the first host in the list. For details, see the `fsclustadm (1m)` manpage.

Of the parameters listed above, `cfs_mount_options` and `cfs_primary_policy` can be edited even when the package is running.

3. Check the package configuration file:

```
cmcheckconf -P /etc/cmcluster/cfspkg1.ascii
```

```
cmcheckconf: Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration
```

4. Apply the package configuration file:

```
cmapplyconf -P /etc/cmcluster/cfspkg1.ascii
```

```
Modify the package configuration ([y]/n)? y
Completed the cluster update
```

5. Run the package:

   cmrunpkg cfspkg1

6. Verify that the modular package of disk group(s) and mount point(s) is running. Also, verify that the file system is mounted:

   cmviewcl

```
Cluster1   up

  NODE           STATUS       STATE
  node1          up           running
  node2          up           running
  node3          up           running
  node4          up           running

MULTI_NODE_PACKAGES

  PACKAGE            STATUS           STATE            AUTO_RUN     SYSTEM
  SG-CFS-pkg         up               running          enabled      yes
  cfspkg1            up               running          enabled      no
```

   bdf

```
Filesystem                kbytes    used     avail    %used   Mounted on
/dev/vg00/lvol3          24215552 20954099 3058098    87%    /
/dev/vg00/lvol1          2031616  365773   1561780    19%    /stand
/dev/odm                    0        0         0         0%    /dev/odm
/dev/vx/dsk/cvm_dg1_node1/lvol1 167808    5313  152346      3%    /mnt2
```

Parallel Activation of Disk Groups and Parallel Mounting of Mount Points

To enable parallel activation of CVM disk groups and parallel mounting of CFS mount points, you must set the following parameters in the package configuration file:

- **cvm_concurrent_dg_operations**

  Specify the number of concurrent disk groups to be activated in parallel during package startup or shutdown. The default value is 1. Setting this value to an appropriate number improves performance when the package needs to activate a large number of disk groups, since the disk groups are activated in parallel thus consuming lesser time. For example, if a package uses three disk groups that need to be activated in parallel, you must enter the following in the package configuration file:

```
cvm_concurrent_dg_operations   3
```

- **cfs_concurrent_mount_unmount_operations**

  Specify the number of concurrent mount operations to allow during package startup or shutdown. The default value is 1. Setting this value to an appropriate number improves performance when the package needs to mount a number of mount points in parallel. For example, if a package uses three disk groups on which the file systems need to be mounted in parallel, you must enter the following in the package configuration file:

```
cfs_concurrent_mount_unmount_operations   3
```

IMPORTANT: It is recommended to increment the parameter values in smaller numbers and monitor the effect on performance at each level. As soon as the performance starts to decline, stop increasing and reset it to a lower level. However, you must consider the factors like the number of CPUs, the amount of available memory, the HP-UX kernel settings, and the number and characteristics of other packages that will be running on the node.

NOTE: Serviceguard does not support parallel deactivation of disk groups and parallel unmounting of mount points. These operations are done serially.

Creating Modular Checkpoint and Snapshot Packages for CFS

Mount Point Packages for Storage Checkpoint using Modular Style Packages

The following example illustrates how to create a storage checkpoint:

1. Create a package configuration file for the storage checkpoint:

```
cmmakepkg -m sg/cfs_all /etc/cmcluster/ckpt1.ascii
Package template is created. This file must be edited before it can be used.
```

2. Edit the following package parameters in the `ckpt1.ascii` package configuration file:

```
package_name                    ckpt1

node_name                       node1
node_name                       node2

ckpt_mount_point                /ckpt1
ckpt_name                       check1
ckpt_cfs_mount_point            /mnt2
ckpt_mount_options              "node1=cluster node2=cluster"
ckpt_primary_policy              node1 node2
```

NOTE: By default, the package parameters are commented out. You must uncomment them and specify the values.

ckpt_mount_point	name of the mount point for the storage checkpoint
ckpt_name	name of the storage checkpoint
ckpt_cfs_mount_point	unique name for CFS mount point; supports up to two levels of nested mount points
ckpt_mount_options	the CFS mount options for the storage checkpoint. For detailed information about the mount options, see the `mount_vxfs (1m)` manpage.
ckpt_primary_policy	sets the primary migration policy for the mount point. This option sets up the order in which hosts assume the cluster file system primary ownership if the primary node fails. If no host specified in the list is active when the primary node fails, a node is chosen at random to be the new primary. When the policy is set, if the first host name specified in the list is not the current primary, a primary migration is triggered to the first host in the list. For details, see the `fsclustadm (1m)` manpage.

Of the parameters listed above, `ckpt_primary_policy` can be edited even when the package is running.

3. Configure the appropriate dependency parameters of the checkpoint:

```
dependency_name                 cfspkg1
dependency_condition            cfspkg1 = UP
dependency_location             same_node
```

4. Check the package configuration file:

   cmcheckconf -P /etc/cmcluster/ckpt1.ascii

   ```
   cmcheckconf: Verification completed with no errors found.
   Use the cmapplyconf command to apply the configuration
   ```

5. Apply the package configuration file:

   cmapplyconf -P /etc/cmcluster/ckpt1.ascii

   ```
   Modify the package configuration ([y]/n)? y
   Completed the cluster update
   ```

6. Run the checkpoint package:

   cmrunpkg ckpt1

   ```
   Running package ckpt1 on node node1
   Running package ckpt1 on node node2
   Successfully started package ckpt1 on node node1 Successfully started package ckpt1 on node node2
   cmrunpkg: All specified packages are running
   ```

7. Verify the modular package of mount point for the checkpoint package(s) running. Also, verify that the file system is mounted:

   cmviewcl

   ```
   CLUSTER          STATUS
   cluster1     up

   NODE            STATUS       STATE
   node1           up           running
   node2           up           running

   MULTI_NODE_PACKAGES

   PACKAGE         STATUS       STATE            AUTO_RUN    SYSTEM
   SG-CFS-pkg      up           running          enabled     yes
   cfspkg1         up           running          enabled     no
   ckpt1           up           running          enabled     no
   ```

Mount Point Packages for Snapshot Images using Modular Style Packages

The following example illustrates how to create a snapshot image. Consider that `cvm_dg3` is the disk group configured into a package `dg2pkg`.

1. Create local storage on which to place the snapshot:

   vxdg init cvm_dg3 c4t1d0

   vxassist -g cvm_dg3 make vol1 100m

   vxvol -g cvm_dg3 startall

2. Create a package configuration file for the snapshot image:

   cmmakepkg -m sg/cfs_all snap1.ascii

   ```
   Package template is created. This file must be edited before it can be used.
   ```

3. Edit the following package parameters in the `snap1.ascii` package configuration file:

   ```
   package_name                    snap1

   node_name                      node1

   snapshot_mount_point           /snap1
   snapshot_name                  cvm_dg3_node1/lvol3
   snapshot_cfs_mount_point       /mnt2
   snapshot_mount_options         "node1=ro"
   ```

NOTE: By default, the package parameters are commented out. You must uncomment them and specify the values.

snapshot_mount_point	name of the mount point for the snapshot
snapshot_name	Diskgroups/Logical volume pair used by the snapshot
snapshot_cfs_mount_point	unique name for CFS mount point; supports up to two levels of nested mount points
snapshot_mount_options	the CFS mount options for the snapshot. For detailed information about the mount options, see the `mount_vxfs (1m)` manpage.

4. Configure the appropriate dependency parameters of the snapshot image:

```
dependency_name             dg2pkg
dependency_condition        dg2pkg = UP
dependency_location         same_node
```

5. Check the package configuration file:

```
cmcheckconf -P snap1.ascii
```

```
cmcheckconf: Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration
```

6. Apply the packages configuration file:

```
cmapplyconf -P snap1.ascii
```

```
Modify the package configuration ([y]/n)? y
Completed the cluster update
```

7. Run the snapshot package:

```
cmrunpkg snap1
```

```
Running package snap1 on node node1
Successfully started package snap1 on node node1
cmrunpkg: All specified packages are running
```

8. Verify the modular package of the mount point for the snapshot(s) running. Also, verify that the file system is mounted:

```
cmviewcl
```

```
CLUSTER         STATUS
cluster1     up

  NODE           STATUS       STATE
  node1          up           running
  node2          up           running

MULTI_NODE_PACKAGES

  PACKAGE        STATUS       STATE            AUTO_RUN       SYSTEM
  SG-CFS-pkg     up           running          enabled          yes
  dg2pkg         up           running          enabled          no
  snap1          up           running          enabled          no
```

```
bdf
```

```
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3    2686976  629944 1928574   25% /
/dev/vg00/lvol1    2031616  361600 1565705   19% /stand
/dev/vg00/lvol5    32636928 1917471 28805207    6% /var
/dev/vg00/lvol7    10584064 3269339 6857606   32% /usr
```

```
/dev/vg00/lvol4     2048000   18290 1902964    1% /tmp
/dev/vg00/lvol6     10076160 5666019 4134547   58% /opt
/dev/vg00/lvol8     3063808   17747 2855686    1% /home
/dev/odm                  0       0       0    0% /dev/odm
/dev/vx/dsk/cvm_dg3_node1/lvol3
                     167776    3260  154234    2% /snap1
```

Guidelines for Migrating from Legacy CFS Package to Modular CFS Package

- You must migrate the legacy CFS packages to modular CFS package only when the package is offline.
- It is recommended that all the disk groups and mount points used by an application in a package be consolidated into one single modular package so that the dependencies need not be configured explicitly in the configuration files. It is also recommended not to share the consolidated modular package between applications unless the applications are interdependent.
- A mount point in the consolidated modular package must have its disk groups in the same package and vice versa.
- It is recommended that you do not configure any inter-dependencies between modular CFS packages and legacy CFS packages.

  NOTE: Legacy CFS packages cannot be configured to depend on a modular CFS package. This is not supported either by command line interpreter or via VEA.

- It is not recommended to configure the storage belonging to different applications into one modular package.
- All the disk group and mount point packages that are consolidated into the modular package must be configured on the same set of nodes in the cluster.
- The storage checkpoint and snapshot packages cannot be merged with other general purpose disk group and mount point packages. It is recommended that all checkpoint and snapshot packages are merged into individual modular checkpoint and snapshot packages respectively. However, configuring diskgroups in snapshot packages is allowed to enable consolidation of storage required by snapshots in the same package.
- When you have Metrocluster for Oracle RAC clusters on different sites, the same package configuration must be present on all the sites for modular CFS packages. In other words, there should be an exact mirror copy of the package configuration on all the sites in a Metrocluster for Oracle RAC cluster environment.

Use Case Scenarios — Migrating Legacy CFS Packages to Modular CFS Packages

Use Case 1: Migration of application using two disk groups

Figure 37 depicts the legacy style of packaging where a separate package is created for each disk group and mount point in the cluster. The mount points, SG-CFS-MP-1 and SG-CFS-MP-2, are created on the cvm_dg1 and cvm_dg2 disk groups respectively, and the cluster file systems /cfs1 and /cfs2 are mounted on the mount points. Application 1 is dependent on two mount point packages in the cluster.

Figure 37 Legacy Style of Packaging

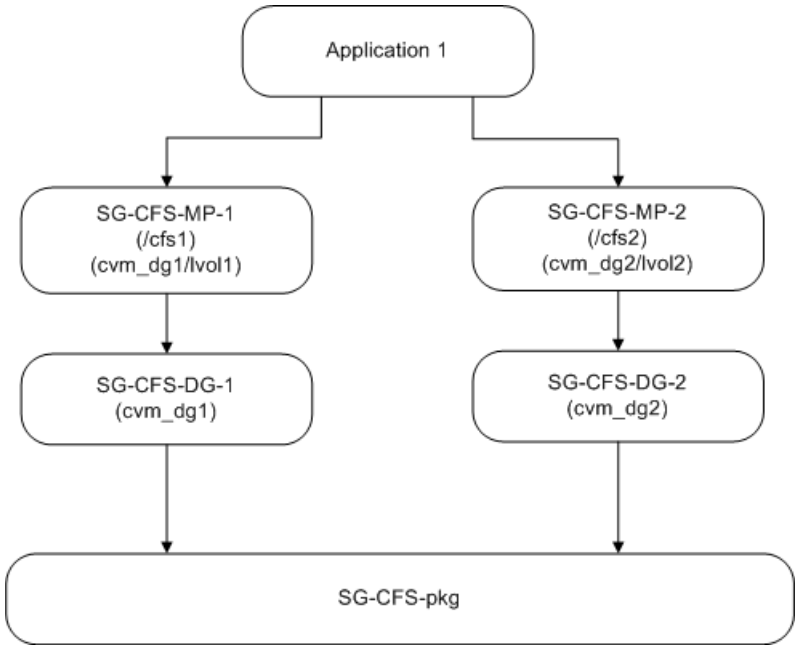
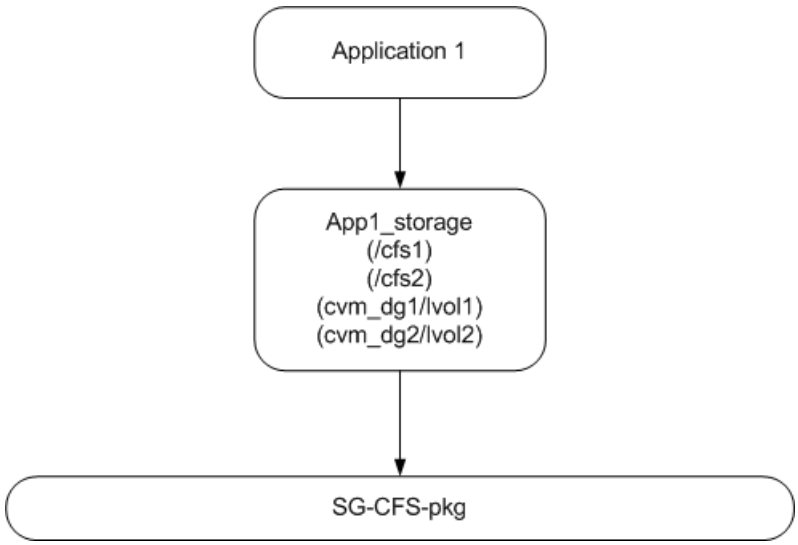


Figure 38 illustrates an example in which all the disk group and mount point packages used by Application 1 are consolidated into a single modular package as compared to four separate packages in the legacy style of packaging.

Figure 38 Modular Style of Packaging



Use Case 2: Migration of two different applications using checkpoint and snapshot packages

Figure 39 illustrates a cluster with two applications independent of each other. The mount points, SG-CFS-MP-1 and SG-CFS-MP-2, are created on the cvm_dg1 and cvm_dg2 disk groups with two logical volumes, lvol1 and lvol2, respectively. The cluster file systems /mnt1 and /mnt2 are mounted on the mount points. Note that the storage check point package, SG-CFS-CK-1, and the snapshot package SG-CFS-SN-1 have a dependency on the mount point package.

Figure 39 Legacy Style of Packaging

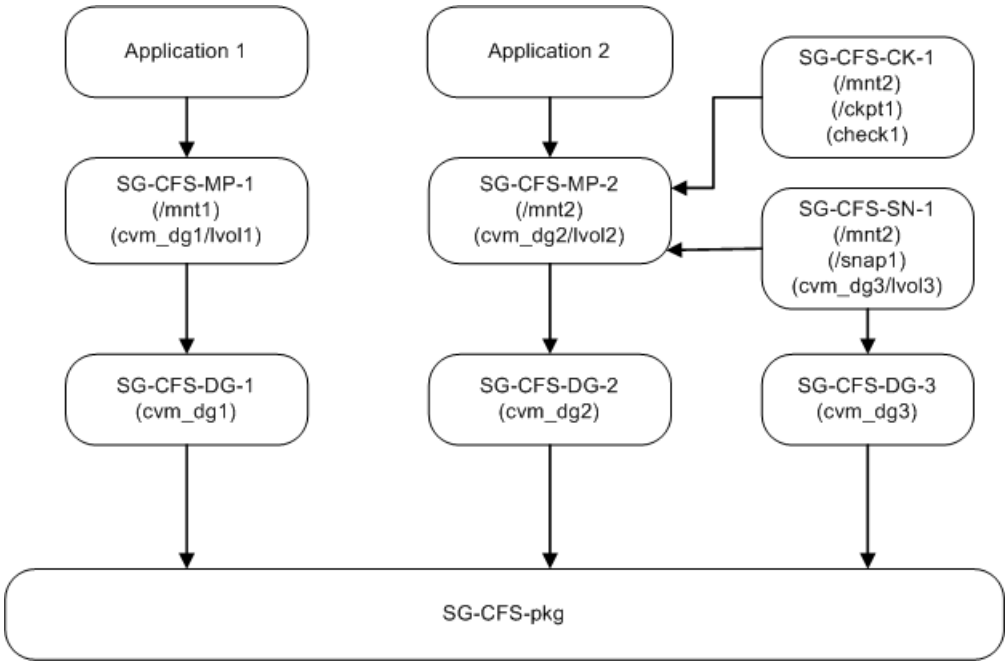
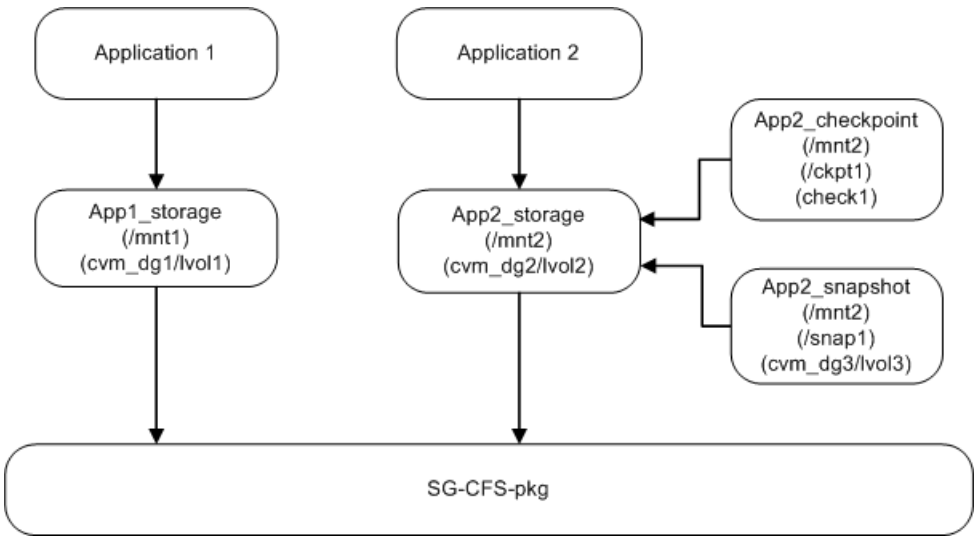


Figure 40 illustrates the consolidated modular CFS packages; all the disk group and mount point packages used by Application 1 are consolidated into a single modular package. Similarly, the disk group and mount point packages used by Application 2 are consolidated into a single modular package. The disk group, cvm_dg3, is merged with the snapshot package to enable consolidation of storage within the same package. The checkpoint package is configured into a single package as it cannot have disk groups, mount points, or snapshots configured within the same package. This consolidation has reduced the number of packages to six in the modular style of packaging as compared with the legacy style of packaging where the number of packages is nine. It is recommended to consolidate the disk group and mount point packages into modular packages for Application 1 and Application 2 separately, and not to share the modular packages between the applications unless the applications are interdependent.

Figure 40 Modular Style of Packaging



For steps on how to migrate from legacy CFS packages to modular CFS packages, see Appendix G "Migrating from Legacy CFS Packages to Modular CFS Packages " (page 363)

Managing Disk Groups and Mount Points Using Legacy Packages

To manage disk groups and mount points in a Serviceguard cluster using CVM/CFS, create disk group and mount point packages. These packages are configured for activating disk groups and mounting the volumes. The applications that use CVM/CFS require the disk group and mount point packages to be up and running.

Creating a File System and Mount Point Package

1. Create a file system:

   `newfs -F vxfs /dev/vx/rdsk/logdata/log_files`

   ```
   version 6 layout
   1-048576 sectors, 1048576 blocks of size 1024, log size 16384 blocks
   largefiles supported
   ```

2. Create the cluster mount point:

   `cfsmntadm add logdata log_files /tmp/logdata/log_files all=rw`

   ```
   Package name "SG-CFS-MP-1" is generated to control the resource.
   ```

   You do not need to create the directory. The command creates one on each of the nodes, during the mount.

3. Verify with `cmviewcl` or `cfsmntadm display`. This example uses the `cfsmntadm` command:

   `cfsmntadm display`

   ```
   Cluster Configuration for Node: ftsys9
   MOUNT POINT            TYPE    SHARED VOLUME   DISK GROUP    STATUS
   /tmp/logdata/log_files  regular  log_files      logdata       NOT MOUNTED

   Cluster Configuration for Node: ftsys10
   MOUNT POINT            TYPE    SHARED VOLUME   DISK GROUP    STATUS
   /tmp/logdata/log_files  regular  log_files      logdata       NOT MOUNTED
   ```

4. Mount the file system:

   `cfsmount /tmp/logdata/log_files`

   This starts up the multi-node package and mounts a cluster-wide file system.

5. Verify that multi-node package is running and file system is mounted:

   `cmviewcl`

   ```
   CLUSTER   STATUS
   cfs_cluster   up
   NODE   STATUS   STATE
   ftsys9   up     running
   ftsys10  up     running

   MULTI_NODE_PACKAGES
   PACKAGE      STATUS     STATE     AUTO_RUN    SYSTEM
   SG-CFS-pkg   up         running    enabled     yes
   SG-CFS-DG-1  up         running    enabled     no
   SG-CFS-MP-1  up         running    enabled     no
   ```

   ```
   ftsys9/etc/cmcluster/cfs> bdf
   Filesystem                    kbytes used avail  %used  Mounted on
   /dev/vx/dsk/logdata/log_files  10485 17338 966793   2%     tmp/logdata/log_files
   ftsys10/etc/cmcluster/cfs> bdf
   Filesystem                    kbytes used avail  %used  Mounted on
   /dev/vx/dsk/logdata/log_files  10485 17338 966793   2%     tmp/logdata/log_files
   ```

6. To view the package name that is monitoring a mount point, use the `cfsmntadm show_package` command:

   `cfsmntadm show_package /tmp/logdata/log_files`

```
SG-CFS-MP-1
```

7. After creating the mount point packages for the cluster file system, configure your application package to depend on the mount points. In the configuration file, configure a dependency, setting *dependency_condition* to `SG-CFS-MP-pkg-# =UP` and *dependency_location* to `SAME_NODE`. For more information about these parameters, see "Package Parameter Explanations" (page 222).

Creating Checkpoint and Snapshot Packages for CFS

The storage checkpoints and snapshots are two additional mount point package types. They can be associated with the cluster via the `cfsmntadm`(1m) command.

Mount Point Packages for Storage Checkpoints using Legacy Style Packages

The Veritas File System provides a unique storage checkpoint facility which quickly creates a persistent image of a file system at an exact point in time. Storage checkpoints significantly reduce I/O overhead by identifying and maintaining only the file system blocks that have changed since the last storage checkpoint or backup. This is done by a copy-on-write technique. Unlike a disk-based mirroring technology, which requires a separate storage space, this Veritas technology minimizes the use of disk space by creating a storage checkpoint within the same free space available to the file system.

For more information about the technique, see the Veritas File System Administrator's Guide appropriate to your version of CFS, posted at http://www.hp.com/go/hpux-core-docs.

The following example illustrates how to create a storage checkpoint of the `/cfs/mnt2` file system.

Start with a cluster-mounted file system.

1. Create a checkpoint of `/tmp/logdata/log_files` named `check2`. It is recommended that the file system already be part of a mount point package that is mounted.

```
cfsmntadm display
Cluster Configuration for Node: ftsys9
MOUNT POINT               TYPE      SHARED VOLUME    DISK GROUP     STATUS
  /tmp/logdata/log_files   regular   log_files        logdata        MOUNTED

Cluster Configuration for Node: ftsys10
   MOUNT POINT               TYPE     SHARED VOLUME    DISK GROUP     STATUS
   /tmp/logdata/log_files    regular  log_files        logdata        MOUNTED
fsckptadm -n create check2 /tmp/logdata/log_files
```

2. Associate it with the cluster and mount it.

```
cfsmntadm add ckpt check2 /tmp/logdata/log_files \
/tmp/check_logfiles all=rw
Package name "SG-CFS-CK-2" was generated to control the resource
Mount point "/tmp/check_logfiles" was associated to the cluster
cfsmount /tmp/check_logfiles
```

3. Verify:

```
cmviewcl
CLUSTER      STATUS
cfs-cluster  up

   NODE          STATUS       STATE
   ftsys9        up           running
   ftsys10       up           running

MULTI_NODE_PACKAGES

   PACKAGE       STATUS        STATE         AUTO_RUN      SYSTEM
   SG-CFS-pkg    up            running       enabled       yes
```

```
   SG-CFS-DG-1  up           running      enabled      no
   SG-CFS-MP-1  up           running      enabled      no
   SG-CFS-CK-1  up           running      disabled     no
```

`/tmp/check_logfiles` now contains a point-in-time view of `/tmp/logdata/log_files`, and it is persistent.

```
bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     544768  352240  180540   66% /
/dev/vg00/lvol1     307157   80196  196245   29% /stand
/dev/vg00/lvol5    1101824  678124  398216   63% /var
/dev/vg00/lvol7    2621440 1702848  861206   66% /usr
/dev/vg00/lvol4       4096     707    3235   18% /tmp
/dev/vg00/lvol6    2367488 1718101  608857   74% /opt
/dev/vghome/varopt 4194304  258655 3689698    7% /var/opt
/dev/vghome/home   2097152   17167 1949993    1% /home
/tmp/logdata/log_files
                    102400    1898   94228    2% /tmp/logdata/log_files
/tmp/logdata/log_files:check2
                    102400    1898   94228    2% /tmp/check_logfiles
```

Mount Point Packages for Snapshot Images using Legacy Style Packages

A snapshot is a frozen image of an active file system that does not change when the contents of target file system changes. On cluster file systems, snapshots can be created on any node in the cluster, and backup operations can be performed from that node. The snapshot of a cluster file system is accessible only on the node where it is created; the snapshot file system itself cannot be cluster mounted.

For details on creating snapshots on cluster file systems, see the *Veritas Storage Foundation Cluster File System Installation and Administration Guide* posted at http://www.hp.com/go/hpux-serviceguard-docs.

The following example illustrates how to create a snapshot of the `/tmp/logdata/log_files` file system.

1. Create local storage on which to place the snapshot.

   ```
   vxdg init dg1 c4t1d0
   vxassist -g dg1 make vol1 100m
   vxvol -g dg1 startall
   ```

2. Associate it with the cluster.

   ```
   cfsmntadm add snapshot dev=/dev/vx/dsk/dg1/vol1 \
   /tmp/logdata/log_files /local/snap1 ftsys9=ro
   Package name "SG-CFS-SN-1" was generated to control the resource
   Mount point "/local/snap1" was associated to the cluster
   cfsmount /local/snap1
   cmviewcl
   CLUSTER      STATUS
   cfs-cluster  up

      NODE         STATUS       STATE
      ftsys9       up           running
      ftsys10       up           running
   MULTI_NODE_PACKAGES

      PACKAGE      STATUS       STATE        AUTO_RUN     SYSTEM
      SG-CFS-pkg   up           running      enabled      yes
      SG-CFS-DG-1  up           running      enabled      no
   ```

```
   SG-CFS-MP-1  up           running      enabled      no
   SG-CFS-SN-1  up           running      disabled     no
```

The snapshot file system `/local/snap1` is now mounted and provides a point in time view of `/tmp/logdata/log_files`.

`bdf`

```
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     544768  352233  180547   66% /
/dev/vg00/lvol1     307157   80196  196245   29% /stand
/dev/vg00/lvol5    1101824  678426  397916   63% /var
/dev/vg00/lvol7    2621440 1702848  861206   66% /usr
/dev/vg00/lvol4       4096     707    3235   18% /tmp
/dev/vg00/lvol6    2367488 1718101  608857   74% /opt
/dev/vghome/varopt 4194304  258609 3689741    7% /var/opt
/dev/vghome/home   2097152   17167 1949993    1% /home
/dev/vx/dsk/logdata/log_files
                     102400    1765   94353    2% /tmp/logdata/log_files
/dev/vx/dsk/dg1/vol1
                     102400    1765   94346    2% /local/snap1
```

Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information on support for CVM (and CFS, the Cluster File System): http://www.hp.com/go/hpux-serviceguard-docs.

This section explains how to configure the Veritas Cluster Volume Manager *without* Veritas CFS (Cluster File System) — for example, to set up raw devices for Serviceguard Extension for RAC. You can use the instructions below to configure CVM disk groups for raw access whether you have purchased CVM as a standalone product, or as part of a bundle that includes CFS.

NOTE: You *must* use CFS if you want to configure file systems on your CVM disk groups; see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190).

The two sets of procedures — with and without CFS — use many of the same commands, but in a slightly different order.

Before you start, 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, `VEA`, to carry out configuration tasks. Instructions for running `vxinstall` are in the Veritas Installation Guide for your version. For more information, refer to the Veritas Volume Manager Administrator's Guide for your version.

You need to do the tasks described in the following sections:

- "Initializing the Veritas Volume Manager " (page 209)
- "Preparing the Cluster for Use with CVM " (page 209)
- "Identifying the Master Node" (page 209)
- "Initializing Disks for CVM" (page 209)
- "Creating Disk Groups" (page 210)
- "Creating Volumes " (page 210)
- "Adding Disk Groups to the Package Configuration " (page 210)

For more information, including details about configuration of plexes (mirrors), multipathing, and RAID, refer to the HP-UX documentation for the Veritas Volume Manager. See the documents for HP Serviceguard Storage Management Suite posted at http://www.hp.com/go/hpux-serviceguard-docs.

Initializing the Veritas Volume Manager

If you are about to create disk groups for the first time, you need to initialize the Volume Manager.

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.

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 Serviceguard-supplied CVM system multi-node package, `SG-CFS-pkg`. This means that the cluster must already be configured and running before you create disk groups.

NOTE: Cluster configuration is described under "Configuring the Cluster " (page 177).

Check the heartbeat configuration. CVM 4.1 and later versions require that the cluster have either multiple heartbeats or a single heartbeat with a standby, and do not allow the use of Auto Port Aggregation, Infiniband, or VLAN interfaces as a heartbeat subnet.

The CVM cluster volumes are managed by a Serviceguard-supplied system multi-node package which runs on all nodes at once, and cannot fail over. For CVM 4.1 and later, Serviceguard creates the `SG-CFS-pkg`.

The `SG-CFS-pkg` package has the following responsibilities:

- Maintain Veritas configuration files `/etc/llttab`, `/etc/llthosts`, `/etc/gabtab`
- Launch required services: `cmvxd`, `cmvxpingd`, `vxfsckd`
- Start/halt Veritas process in the proper order: `llt`, `gab`, `vxfen`, `odm`, `cvm`, `cfs`

Use the `cmapplyconf` command to create the system multi-node package that communicates cluster information to CVM:

```
cmapplyconf -P /etc/cmcluster/cfs/SG-CFS-pkg.conf
Begin package verification ...
Modify the package configuration ([y]/n)?  Y
Completed the cluster update
```

IMPORTANT: Do this *only* if you are using CVM without CFS. If you *are* using CFS, you set up CVM as part of CFS; see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190).

Identifying the Master Node

If it is not already running, start the cluster. This will automatically 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 c4t3d4
```

Creating Disk Groups

Use the following steps 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

   logdata      enabled,shared       972078742.1084.node2
   ```

3. Activate the disk group, as follows, before creating volumes:

   ```
   vxdg -g logdata set activation=sw
   ```

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.

The global policy is recommended, because it ensures all nodes are accessing the same current data. If you use local, it can cause problems if one node cannot update one of the mirror copies and the data on that copy goes stale. If any of the other nodes read from that mirror copy, they will read stale data. This can be avoided with the global option, because all nodes will only use the current mirror copy, so they will all read consistent data.

This policy can be reset 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 multipath storage using CVM are described in the HP-UX documentation for the Veritas Volume Manager, posted at http://www.hp.com/go/hpux-core-docs.

Creating Volumes

Use the `vxassist` command to create 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
```

Now deport the disk groups:

```
vxdg deport <disk group name>
```

Adding Disk Groups to the Package Configuration

Next you need to specify the CVM disk groups for each package that uses them. You do this in the package configuration file for modular packages and in the package control script for legacy

packages. Use one *cvm_dg* for each disk group the package will use, and select the appropriate *cvm_activation_cmd*. Package configuration is described in detail in Chapter 6 (Chapter 7 for legacy packages).

All packages (both legacy and modular) that use CVM disk groups also need to declare a dependency on the `SG-CFS-pkg` in their package configuration file, specifying *dependency_condition* as `UP` and *dependency_location* as `same_node`; for example:

```
dependency_name SG-CFS-pkg_dep
dependency_condition SG-CFS-pkg = UP
dependency_location same_node
```

Using DSAU during Configuration

You can use DSAU to centralize and simplify configuration and monitoring tasks. See "What are the Distributed Systems Administration Utilities?" (page 24).

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

You can check configuration and status information using Serviceguard Manager: from the System Management Homepage (SMH), choose `Tools-> Serviceguard Manager`.

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.
- On systems that support CFS, the Veritas Cluster File System, `cfscluster status` gives information about the cluster;`cfsdgadm` display gives information about the cluster's disk groups.
- `cmrunnode` is used to start Serviceguard on 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 user 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 running, 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. you can use Serviceguard Manager to do this, or use the command `cmviewcl -v`.

   Make sure that all nodes and networks are functioning as expected. For more information, see Chapter 7: "Cluster and Package Maintenance" (page 248).
3. Verify that nodes leave and enter the cluster as expected using the following steps:
   - Halt the node. You can use Serviceguard Manager or use the `cmhaltnode` command.
   - Check the cluster membership to verify that the node has left the cluster. You can do this in Serviceguard Manager, or use the `cmviewcl` command.
   - Start the node. You can use Serviceguard Manager or use the `cmrunnode` command.
   - To verify that the node has returned to operation, check in Serviceguard Manager, or use the `cmviewcl` command again.
4. Bring down the cluster. You can do this in Serviceguard Manager, or use the `cmhaltcl -v -f` command.

Additional cluster testing is described in See "Troubleshooting Your Cluster" (page 308). 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 LVM 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. One way to ensure that this does not happen is to edit the `/etc/lvmrc` file on all nodes, setting `AUTO_VG_ACTIVATE` to 0, then including 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: Special considerations apply in the case of the root volume group:

- If the root volume group is mirrored using MirrorDisk/UX, include it in the `custom_vg_activation` function so that any stale extents in the mirror will be re-synchronized.
- Otherwise, the root volume group does not need to be included in the `custom_vg_activation` function, because 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 the other two cases becomes true in that time, startup will fail.

To enable automatic cluster start, set the flag *AUTOSTART_CMCLD* to 1 in `/etc/rc.config.d/cmcluster`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
```

NOTE: The `/sbin/init.d/cmcluster` file may call files that Serviceguard stores in `/etc/cmcluster/rc`. This directory is for Serviceguard use only! Do not move, delete, modify, or add files in this directory.

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 want 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 file, specify all the LANs that you want Serviceguard to monitor. Use the *STATIONARY_IP* parameter, rather than *HEARTBEAT_IP*, to specify LANs that already have IP addresses. For standby LANs, all that is required is the *NETWORK_INTERFACE* parameter 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 rejoin 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 system reset, 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.

Disabling identd

Ignore this section unless you have a particular need to disable `identd`.

You can configure Serviceguard not to use `identd`.

△ **CAUTION:** This is not recommended. Disabling `identd` removes an important security layer from Serviceguard. See the white paper *Securing Serviceguard* at http://www.hp.com/go/hpux-serviceguard-docs for more information.

If you must disable `identd`, you can do so by adding the `-i` option to the `tcp hacl-cfg` and `hacl-probe` commands in `/etc/inetd.conf`.

For example:

1. Change the `cmclconfd` entry in `/etc/inetd.conf` to:

   ```
   hacl-cfg stream tcp nowait root /usr/lbin/cmclconfd cmclconfd -c -i
   ```

2. Change the `cmomd` entry in `/etc/inetd.conf` to (all on one line):

   ```
   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
   ```

Deleting the Cluster Configuration

As root user, you can delete a cluster configuration from all cluster nodes by using Serviceguard Manager or the command line. The `cmdeleteconf` command prompts for a verification before deleting the files unless you use the `-f` option. You can delete the configuration only 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

Serviceguard packages group together applications and the services and resources they depend on.

The typical Serviceguard package is a failover package that starts on one node but can be moved ("failed over") to another if necessary. See "What is Serviceguard? " (page 20), "How the Package Manager Works" (page 48), and "Package Configuration Planning " (page 120) for more information.

You can also create multi-node packages, which run on more than one node at the same time.

System multi-node packages, which run on all the nodes in the cluster, are supported only for applications supplied by HP.

Before you begin, make sure you have done the necessary planning; see "Package Configuration Planning " (page 120).

Creating or modifying a package requires the following broad steps, each of which is described in the sections that follow:

1. Decide on the package's major characteristics and choose the modules you need to include ((page 217)).
2. Generate the package configuration file (page 241).
3. Edit the configuration file (page 242).
4. Verify and apply the package configuration (page 245).
5. Add the package to the cluster (page 246).

NOTE: This is a new process for configuring packages, as of Serviceguard A.11.18. This manual refers to packages created by this method as modular packages, and assumes that you will use it to create new packages; it is simpler and more efficient than the older method, allowing you to build packages from smaller modules, and eliminating the separate package control script and the need to distribute it manually.

Packages created using Serviceguard A.11.17 or earlier are referred to as legacy packages. If you need to reconfigure a legacy package (rather than create a new package), see "Configuring a Legacy Package" (page 289).

It is also still possible to create new legacy packages by the method described in "Configuring a Legacy Package". If you are using a Serviceguard Toolkit such as Serviceguard NFS Toolkit, consult the documentation for that product.

If you decide to convert a legacy package to a modular package, see "Migrating a Legacy Package to a Modular Package" (page 297). Do not attempt to convert Serviceguard Toolkit packages.

(Parameters that are in the package control script for legacy packages, but in the package configuration file instead for modular packages, are indicated by (S) in the tables later in this section (page 220).)

Choosing Package Modules

IMPORTANT: Before you start, you need to do the package-planning tasks described under "Package Configuration Planning " (page 120).

To choose the right package modules, you need to decide the following things about the package you are creating:

- What type of package it is; see "Types of Package: Failover, Multi-Node, System Multi-Node" (page 217).
- Which parameters need to be specified for the package (beyond those included in the base type, which is normally failover, multi-node, or system-multi-node). See "Package Modules and Parameters" (page 219).

When you have made these decisions, you are ready to generate the package configuration file; see "Generating the Package Configuration File" (page 241).

Types of Package: Failover, Multi-Node, System Multi-Node

There are three types of packages:

- Failover packages. This is the most common type of package. Failover packages run on one node at a time. If there is a failure, Serviceguard (or a user) can halt them, and then start them up on another node selected from the package's configuration list; see "*node_name* (page 223)".

  To generate a package configuration file that creates a failover package, include `-m sg/failover` on the `cmmakepkg` command line. See "Generating the Package Configuration File" (page 241).

- Multi-node packages. These packages run simultaneously on more than one node in the cluster. Failures of package components such as applications, services, EMS resources, or subnets, will cause the package to be halted only on the node on which the failure occurred.

  The Veritas Cluster File System (CFS) system multi-node packages are examples of multi-node packages; but support for multi-node packages is no longer restricted to CVM/CFS; you can create a multi-node package for any purpose.

IMPORTANT: If the package uses volume groups, they must be activated in shared mode: `vgchange -a s`.

NOTE: On systems that support CFS, you configure the legacy CFS multi-node packages by means of the `cfsdgadm` and `cfsmntadm` commands, not by editing a package configuration file. For modular CFS packages, you cannot edit using the cfs commands, but must edit the parameters in the package configuration files; see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190).

Note that relocatable IP addresses cannot be assigned to multi-node packages. See also "Differences between Failover and Multi-Node Packages" (page 218).

To generate a package configuration file that creates a multi-node package, include `-m sg/multi_node` on the `cmmakepkg` command line. See "Generating the Package Configuration File" (page 241).

- System multi-node packages. These packages run simultaneously on every node in the cluster. They cannot be started and halted on individual nodes.

  Both *`node_fail_fast_enabled`* (page 224) and *`auto_run`* (page 224) must be set to `yes` for this type of package. All services must have *`service_fail_fast_enabled`* (page 233) set to `yes`.

  System multi-node packages are supported only for applications supplied by HP, for example Veritas Cluster File System (CFS).

  NOTE: On systems that support CFS, you configure the legacy CFS system multi-node package by means of the `cfscluster` command, not by editing a package configuration file. For modular CFS packages, you cannot edit using the cfs commands, but must edit the parameters in the package configuration files; see "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190).

NOTE: The following parameters cannot be configured for multi-node or system multi-node packages:

- *`failover_policy`*
- *`failback_policy`*
- *`ip_subnet`*
- *`ip_address`*

Volume groups configured for packages of these types must be activated in shared mode.

For more information about types of packages and how they work, see "How the Package Manager Works" (page 48). For information on planning a package, see "Package Configuration Planning " (page 120).

When you have decided on the type of package you want to create, the next step is to decide what additional package-configuration modules you need to include; see "Package Modules and Parameters" (page 219).

Differences between Failover and Multi-Node Packages

Note the following important differences in behavior between multi-node and failover packages:

- If a multi-node package has *`auto_run`* disabled (set to `no` in the package configuration file) it will not start when the cluster is started. You can use `cmmodpkg` to enable package switching

and start the package for the first time. But if you then halt the multi-node package via `cmhaltpkg`, it can be re-started only by means of `cmrunpkg`, not `cmmodpkg`.

- If a multi-node package is halted via `cmhaltpkg`, package switching is not disabled. This means that the halted package will start to run on a rebooted node, if it is configured to run on that node and its dependencies are met.
- When a multi-node package is started the first time (either at cluster startup, or subsequently if *`auto_run`* is set to no, and package switching is then enabled) any dependent package will start on its primary node. But if a multi-node package is halted along with its dependent packages, and the multi-node package is then restarted, dependent packages which have had package switching re-enabled will start on the first eligible node on which an instance of the multi-node package comes up; this may not be the dependent packages' primary node.

  To ensure that dependent failover packages restart on their primary node if the multi-node packages they depend on need to be restarted, make sure the dependent packages' package switching is not re-enabled before the multi-node packages are restarted. You can then either restart the dependent failover packages with `cmrunpkg`, specifying the node you want them to start on, or enable package switching for these packages after the multi-node package startup is complete.

Package Modules and Parameters

The table that follows shows the package modules and the configuration parameters each module includes. Read this section in conjunction with the discussion under "Package Configuration Planning " (page 120).

Use this information, and the parameter explanations that follow ((page 222)), to decide which modules (if any) you need to add to the failover or multi-node module to create your package. If you are used to creating legacy packages, you will notice that parameters from the package control script (or their equivalents) are now in the package configuration file; these parameters are marked (S) in the table.

You can use `cmmakepkg -l` (letter "l") to see a list of all available modules, including non-Serviceguard modules such as those supplied in the HP Toolkits.

NOTE: If you are going to create a complex package that contains many modules, you may want to skip the process of selecting modules, and simply create a configuration file that contains all the modules:

```
cmmakepkg -m sg/all $SGCONF/sg-all
```

(The output will be written to `$SGCONF/sg-all`.)

Base Package Modules

At least one base module (or `default` or `all`, which include the base module) must be specified on the `cmmakepkg` command line. Parameters marked with an asterisk (*) are new or changed as of Serviceguard A.11.18 or A.11.19. (S) indicates that the parameter (or its equivalent) has moved from the package control script to the package configuration file for modular packages. See the "Package Parameter Explanations" (page 222) for more information.

Table 10 Base Modules

Module Name	Parameters (page)	Comments
`failover`	*`package_name`* (page 222) * *`module_name`* (page 223) * *`module_version`* (page 223) *	Base module. Use as primary building block for failover packages.

Table 10 Base Modules *(continued)*

Module Name	Parameters (page)	Comments
	package_type (page 223) *package_description* (page 223) * *node_name* (page 223) *auto_run* (page 224) *node_fail_fast_enabled* (page 224) *run_script_timeout* (page 224) *halt_script_timeout* (page 225) *successor_halt_timeout* (page 225) * *script_log_file* (page 225) *operation_sequence* (page 225) * *log_level* (page 226) * *failover_policy* (page 226) *failback_policy* (page 227) *priority* (page 227) *	Cannot be used if *package_type* (page 223) is `multi_node` or `system_multi_node`.
`multi_node`	*package_name* (page 222) * *module_name* (page 223) * *module_version* (page 223) * *package_type* (page 223) *package_description* (page 223) *node_name* (page 223) * *auto_run* (page 224) *node_fail_fast_enabled* (page 224) *run_script_timeout* (page 224) *halt_script_timeout* (page 225) *successor_halt_timeout* (page 225) * *script_log_file* (page 225) *operation_sequence* (page 225) * *log_level* (page 226) *priority* (page 227) *	Base module. Use as primary building block for multi-node packages. Cannot be used if *package_type* (page 223) is `failover` or `system_multi_node`.
`system_multi_node`	(as `multi_node`)	Base module. Primary building block for system multi-node packages, supported only for applications supplied by HP.

Optional Package Modules

Add optional modules to a base module if you need to configure the functions in question. Parameters marked with an asterisk (*) are new or changed as of Serviceguard A.11.18/19. (S) indicates that the parameter (or its equivalent) has moved from the package control script to the package configuration file for modular packages. See the "Package Parameter Explanations" (page 222) for more information.

Table 11 Optional Modules

Module Name	Parameters (page)	Comments
`dependency`	*dependency_name* (page 227) * *dependency_condition* (page 228) * *dependency_location* (page 228) *	Add to a base module to create a package that depends on one or more other packages.
`weight`	*weight_name* (page 229) * *weight value* (page 229) *	Add to a base module to create a package that has weight that will be counted against a node's capacity.
`monitor_subnet`	*local_lan_failover_allowed* (page 229) *monitored_subnet* (page 229) * *monitored_subnet_access* (page 230) * *cluster_interconnect_subnet* (page 230) *	Add to a base module to configure subnet monitoring for the package.

Table 11 Optional Modules *(continued)*

Module Name	Parameters (page)	Comments
package_ip	*ip_subnet* (page 231) * (S) *ip_subnet_node* (page 231) * *ip_address* (page 232) * (S)	Add to `failover` module to assign relocatable IP addresses to a failover package.
service	*service_name* (page 232) * (S) *service_cmd* (page 232) (S) *service_restart* (page 233) * (S) *service_fail_fast_enabled* (page 233) *service_halt_timeout* (page 233)	Add to a base module to create a package that runs an application or service.
resource	*resource_name* (page 233) * *resource_polling_interval* (page 234) *resource_start* (page 234) *resource_up_value* (page 234)	Add to a base module to create a package that monitors critical resources via the Event Monitoring Service (EMS).
volume_group	*concurrent_vgchange_operations* (page 234) (S) *enable_threaded_vgchange* (page 234) * *vgchange_cmd* (page 235) * (S) *cvm_activation_cmd* (page 235) (S) *vxvol_cmd* (page 235) * (S) *vg* (page 236) (S) *cvm_dg* (page 236) (S) *vxvm_dg* (page 236) (S) *vxvm_dg_retry* (page 236) (S) *deactivation_retry_count* (page 236) (S) *kill_processes_accessing_raw_devices* (page 236) (S)	Add to a base module if the package needs to mount file systems on LVM or VxVM volumes, or uses CVM volumes but not CFS.
filesystem	*concurrent_fsck_operations* (page 237) (S) *concurrent_mount_and_umount_operations* (page 237) (S) *fs_mount_retry_count* (page 237) (S) *fs_umount_retry_count* (page 237) * (S) *fs_name* (page 238) * (S) *fs_directory* (page 238) * (S) *fs_type* (page 238) (S) *fs_mount_opt* (page 239) (S) *fs_umount_opt* (page 239) (S) *fs_fsck_opt* (page 239) (S)	Add to a base module to configure file system options for the package.
pev	*pev_* (page 239) *	Add to a base module to configure environment variables to be passed to an external script.
external_pre	*external_pre_script* (page 239) *	Add to a base module to specify additional programs to be run before volume groups and disk groups are activated while the package is starting and after they are deactivated while the package is halting.
external	*external_script* (page 239) *	Add to a base module to specify additional programs to be run during package start and stopped at package halt time.
acp	*user_name* (page 240) *user_host* (page 240) *user_role* (page 240)	Add to a base module to configure Access Control Policies for the package.

Table 11 Optional Modules *(continued)*

Module Name	Parameters (page)	Comments
`all`	all parameters	Use if you are creating a complex package that requires most or all of the optional parameters; or if you want to see the specifications and comments for all available parameters.
`multi_node_all`	all parameters that can be used by a multi-node package; includes `multi_node`, `dependency`, `monitor_subnet`, `service`, `resource`, `volume_group`, `filesystem`, `pev`, `external_pre`, `external`, and `acp` modules.	Use if you are creating a multi-node package that requires most or all of the optional parameters that are available for this type of package.
`default`	(all parameters)	A symbolic link to the `all` module; used if you do not specify a base module on the `cmmakepkg` command line; see "cmmakepkg Examples" (page 241).

NOTE: The default form for parameter names in the modular package configuration file is lower case; for legacy packages the default is upper case. There are no compatibility issues; Serviceguard is case-insensitive as far as the parameter names are concerned. This manual uses lower case, unless the parameter in question is used only in legacy packages, or the context refers exclusively to such a package.

Package Parameter Explanations

Brief descriptions of the package configuration parameters follow.

NOTE: For more information, see the comments in the editable configuration file output by the `cmmakepkg` command, and the `cmmakepkg` manpage.

If you are going to browse these explanations deciding which parameters you need, you may want to generate and print out a configuration file that has the comments for all of the parameters; you can create such a file as follows:

```
cmmakepkg -m sg/all $SGCONF/sg-all
```

or simply

```
cmmakepkg $SGCONF/sg-all
```

This creates a file `/etc/cmcluster/sg-all` that contains all the parameters and comments.

More detailed instructions for running `cmmakepkg` are in the next section, "Generating the Package Configuration File" (page 241). See also "Package Configuration Planning " (page 120).

package_name

Any name, up to a maximum of 39 characters, that:

- starts and ends with an alphanumeric character
- otherwise contains only alphanumeric characters or dot (.), dash (-), or underscore (_)
- is unique among package names in this cluster

IMPORTANT: Restrictions on package names in previous Serviceguard releases were less stringent. Packages whose names do not conform to the above rules will continue to run, but if you reconfigure them, you will need to change the name; `cmcheckconf` and `cmapplyconf` will enforce the new rules.

`module_name`

The module name (for example, `failover`, `service`, etc.) Do not change it. Used in the form of a relative path (for example `sg/failover`) as a parameter to `cmmakepkg` to specify modules to be used in configuring the package. (The files reside in the `$SGCONF/modules` directory; see "Where Serviceguard Files Are Kept" (page 149) for an explanation of Serviceguard directories.)

New for modular packages.

`module_version`

The module version. Do not change it.

New for modular packages.

`package_type`

The type can be `failover`, `multi_node`, or `system_multi_node`. Default is `failover`. You can configure only failover or multi-node packages; see "Types of Package: Failover, Multi-Node, System Multi-Node" (page 217).

Packages of one type cannot include the base module for another; for example, if *`package_type`* is `failover`, the package cannot include the `multi_node`, or `system_multi_node` module.

`package_description`

The application that the package runs. This is a descriptive parameter that can be set to any value you choose, up to a maximum of 80 characters. Default value is `Serviceguard Package`. New for 11.19

`node_name`

The node on which this package can run, or a list of nodes in order of priority, or an asterisk (`*`) to indicate all nodes. The default is `*`.

For system multi-node packages, you must specify `*`.

If you use a list, specify each node on a new line, preceded by the literal `node_name`, for example:

```
node_name <node1>
node_name <node2>
node_name <node3>
```

The order in which you specify the node names is important. First list the primary node name (the node where you normally want the package to start), then the first adoptive node name (the best candidate for failover), then the second adoptive node name, followed by additional node names in order of preference.

In case of a failover, control of the package will be transferred to the next adoptive node name listed in the package configuration file, or (if that node is not available or cannot run the package at that time) to the next node in the list, and so on.

IMPORTANT: See "Cluster Configuration Parameters " (page 105) for important information about node names.

See "About Cross-Subnet Failover" (page 145) for considerations affecting cross-subnet packages, which are further explained under "Cross-Subnet Configurations" (page 29). See "Rules for Simple Dependencies" (page 129) for considerations affecting a package that depends on another package, or that is depended on.

auto_run

Can be set to yes or no. The default is yes.

For failover packages, yes allows Serviceguard to start the package (on the first available node listed under *node_name*) on cluster start-up, and to automatically restart it on an adoptive node if it fails. no prevents Serviceguard from automatically starting the package, and from restarting it on another node.

This is also referred to as package switching, and can be enabled or disabled while the package is running, by means of the cmmodpkg (1m) command.

auto_run should be set to yes if the package depends on another package, or is depended on; see "About Package Dependencies" (page 128).

For system multi-node packages, *auto_run* must be set to yes. In the case of a multi-node package, setting *auto_run* to yes allows an instance to start on a new node joining the cluster; no means it will not.

node_fail_fast_enabled

Can be set to yes or no. The default is no.

yes means the node on which the package is running will be halted (HP-UX system reset) if the package fails; no means Serviceguard will not halt the system.

If this parameter is set to yes and one of the following events occurs, Serviceguard will halt the system (HP-UX system reset) on the node where the control script fails:

- A package subnet fails and no backup network is available
- An EMS resource fails
- Serviceguard is unable to execute the halt function
- The start or halt function times out

NOTE: If the package halt function fails with "exit 1", Serviceguard does not halt the node, but sets no_restart for the package, which disables package switching (*auto_run*), thereby preventing the package from starting on any adoptive node.

Setting *node_fail_fast_enabled* to yes ensures that the package can fail over to another node even if the package cannot halt successfully. Be careful when using *node_fail_fast_enabled*, as it will cause *all* packages on the node to halt abruptly. For more information, see "Responses to Failures " (page 85) and "Responses to Package and Service Failures " (page 87).

For system multi-node packages, *node_fail_fast_enabled* must be set to yes.

run_script_timeout

The amount of time, in seconds, allowed for the package to start; or no_timeout. The default is no_timeout. The maximum is 4294.

If the package does not complete its startup in the time specified by *run_script_timeout*, Serviceguard will terminate it and prevent it from switching to another node. In this case, if *node_fail_fast_enabled* is set to yes, the node will be halted (HP-UX system reset).

If no timeout is specified (no_timeout), Serviceguard will wait indefinitely for the package to start.

If a timeout occurs:

- Switching will be disabled.
- The current node will be disabled from running the package.

NOTE: VxVM disk groups are imported at package run time and exported at package halt time. If a package uses a large number of VxVM disk, the timeout value must be large enough to allow all of them to finish the import or export.

NOTE: If (no_timeout is specified, and the script hangs, or takes a very long time to complete, during the validation step (cmcheckconf (1m)), cmcheckconf will wait 20 minutes to allow the validation to complete before giving up.

halt_script_timeout

The amount of time, in seconds, allowed for the package to halt; or no_timeout. The default is no_timeout. The maximum is 4294.

If the package's halt process does not complete in the time specified by *halt_script_timeout*, Serviceguard will terminate the package and prevent it from switching to another node. In this case, if *node_fail_fast_enabled* (page 224) is set to yes, the node will be halted (HP-UX system reset).

If a *halt_script_timeout* is specified, it should be greater than the sum of all the *values set for service_halt_timeout* (page 233) for this package.

If a timeout occurs:

- Switching will be disabled.
- The current node will be disabled from running the package.

If a halt-script timeout occurs, you may need to perform manual cleanup. See "Package Control Script Hangs or Failures" in Chapter 8. See also the note about VxVM under *run_script_timeout* (page 224).

successor_halt_timeout

Specifies how long, in seconds, Serviceguard will wait for packages that depend on this package to halt, before halting this package. Can be 0 through 4294, or no_timeout. The default is no_timeout.

- no_timeout means that Serviceguard will wait indefinitely for the dependent packages to halt.
- 0 means Serviceguard will not wait for the dependent packages to halt before halting this package.

New as of A.11.18 (for both modular and legacy packages). See also "About Package Dependencies" (page 128).

script_log_file

The full pathname of the package's log file. The default is $SGRUN/log/<*package_name*>.log. (See "Where Serviceguard Files Are Kept" (page 149) for more information about Serviceguard pathnames.) See also *log_level* (page 226).

operation_sequence

Defines the order in which the scripts defined by the package's component modules will start up. See the package configuration file for details.

This parameter is not configurable; do not change the entries in the configuration file.

New for modular packages.

log_level

Determines the amount of information printed to `stdout` when the package is validated, and to the *script_log_file* (page 225) when the package is started and halted. Valid values are `0` through `5`, but you should normally use only the first two (`0` or `1`); the remainder (`2` through `5`) are intended for use by HP Support.

- `0` - informative messages
- `1` - informative messages with slightly more detail
- `2` - messages showing logic flow
- `3` - messages showing detailed data structure information
- `4` - detailed debugging information
- `5` - function call flow

New for modular packages.

failover_policy

Specifies how Serviceguard decides where to restart the package if it fails. Can be set to `configured_node`, `min_package_node`, `site_preferred`, or `site_preferred_manual`. The default is `configured_node`.

- `configured_node` means Serviceguard will attempt to restart the package on the next available node in the list you provide under *node_name* (page 223).
- `min_package_node` means Serviceguard will restart a failed package on whichever node in the *node_name* list has the fewest packages running at the time.
- `site_preferred` means Serviceguard will try all the eligible nodes on the local *SITE* before failing the package over to a node on another *SITE*. This policy can be configured only in a site-aware disaster-tolerant cluster, which requires Metrocluster (additional HP software); see the documents listed under "Cross-Subnet Configurations" (page 29) for more information.
- `site_preferred_manual` means Serviceguard will try to fail the package over to a node on the local *SITE*. If there are no eligible nodes on the local *SITE*, the package will halt with global switching enabled. You can then restart the package locally, when a local node is available, or start it on another *SITE*. This policy can be configured only in a site-aware disaster-tolerant cluster, which requires Metrocluster (additional HP software); see the documents listed under "Cross-Subnet Configurations" (page 29) for more information.

This parameter can be set for failover packages only. For a package that will depend on another package or vice versa, see also "About Package Dependencies" (page 128).

failback_policy

Specifies what action the package manager should take when a failover package is not running on its primary node (the first node on its *node_name* list) and the primary node is once again available. Can be set to automatic or manual. The default is manual.

- manual means the package will continue to run on the current (adoptive) node.
- automatic means Serviceguard will move the package to the primary node as soon as that node becomes available, unless doing so would also force a package with a higher *priority* (page 227) to move.

CAUTION: When the *failback_policy* is automatic and you set the *NODE_NAME* to '*', if you add, delete, or rename a node in the cluster, the primary node for the package might change resulting in the automatic failover of that package.

This parameter can be set for failover packages only. If this package will depend on another package or vice versa, see also "About Package Dependencies" (page 128). If the package has a configured weight, see also "About Package Weights" (page 135).

priority

Assigns a priority to a failover package whose *failover_policy* (page 226) is configured_node. Valid values are 1 through 3000, or no_priority. The default is no_priority. See also the *dependency_* parameter descriptions (page 227).

priority can be used to satisfy dependencies when a package starts, or needs to fail over or fail back: a package with a higher priority than the packages it depends on can drag those packages, forcing them to start or restart on the node it chooses, so that its dependencies are met.

If you assign a priority, it must be unique in this cluster. HP recommends assigning values in increments of 20 so as to leave gaps in the sequence; otherwise you may have to shuffle all the existing priorities when assigning priority to a new package.

IMPORTANT: Because priority is a matter of ranking, a lower number indicates a higher priority (20 is a higher priority than 40). A numerical priority is higher than no_priority.

New for A.11.18 (for both modular and legacy packages). See "About Package Dependencies" (page 128) for more information.

dependency_name

An identifier for a particular dependency that must be met in order for this package to run (or keep running). It must be unique among this package's *dependency_name*s. The length and formal restrictions for the name are the same as for *package_name* (page 222).

IMPORTANT: Restrictions on dependency names in previous Serviceguard releases were less stringent. Packages that specify *dependency_name*s that do not conform to the above rules will continue to run, but if you reconfigure them, you will need to change the *dependency_name*; cmcheckconf and cmapplyconf will enforce the new rules.

Configure this parameter, along with *dependency_condition* and *dependency_location*, if this package depends on another package; for example, if this package depends on a package named pkg2 to be running on the same node:

```
dependency_name pkg2dep  dependency_condition pkg2 = UP  dependency_location same_node
```

For more information about package dependencies, see the parameter descriptions that follow, the cmmakepkg (1m) manpage, and the discussion in this manual "About Package Dependencies" (page 128).

dependency_condition

The condition that must be met for this dependency to be satisfied.

The syntax is: <*package_name*> = <*package_status*>. <*package_name*> is the name of the package depended on. Valid values for <*package_status*> are UP, or DOWN.

- UP means that this package requires the package identified by <*package_name*> to be up (that is, the status reported by cmviewcl is UP).

  If you specify UP, the type and characteristics of the current package (the one you are configuring) impose the following restrictions on the type of package it can depend on:

  - If the current package is a multi-node package, <*package_name*> must identify a multi-node or system multi-node package.
  - If the current package is a failover package and its *failover_policy* (page 226) is min_package_node, <*package_name*> must identify a multi-node or system multi-node package.
  - If the current package is a failover package and its *failover_policy* (page 226) is configured_node , <*package_name*> must identify a multi-node or system multi-node package, or a failover package whose *failover_policy* is configured_node.

- DOWN means that this package requires the package identified by <*package_name*> to be down (that is, the status reported by cmviewcl is UP). This is known as an exclusionary dependency (or exclusion dependency).

  If you specify DOWN:

  - The exclusionary dependency must be mutual: only one of the packages can be running at any given time. For example, if pkgA depends on pkgB to be down, pkgB must also depend on pkgA to be down.
  - You must apply both packages to the cluster configuration at the same time.
  - Both packages must be failover packages whose *failover_policy* is configured_node.
  - At least one of the packages must specify a *priority* (page 227).

For more information, see "About Package Dependencies" (page 128).

dependency_location

Specifies where the *dependency_condition* must be met.

- If *dependency_condition* is UP, legal values for*dependency_location* are same_node, any_node, and different_node.
  - same_node means that the package depended on must be running on the same node.
  - different_node means that the package depended on must be running on a different node in this cluster.
  - any_node means that the package depended on must be running on some node in this cluster.
- If *dependency_condition* is DOWN, legal values for*dependency_location* are same_node and all_nodes.
  - same_node means that the package depended on must not be running on the same node.
  - all_nodes means that the package depended on must not be running on any node in this cluster.

For more information, see "About Package Dependencies" (page 128).

weight_name, weight_value

These parameters specify a weight for a package; this weight is compared to a node's available capacity (defined by the *CAPACITY_NAME* and *CAPACITY_VALUE* parameters in the cluster configuration file) to determine whether the package can run there.

Both parameters are optional, but if *weight_value* is specified, *weight_name* must also be specified, and must come first. You can define up to four weights, corresponding to four different capacities, per cluster. To specify more than one weight for this package, repeat *weight_name* and *weight_value*.

NOTE: But if *weight_name* is `package_limit`, you can use only that one weight and capacity throughout the cluster. `package_limit` is a reserved value, which, if used, must be entered exactly in that form. It provides the simplest way of managing weights and capacities; see "Simple Method" (page 136) for more information.

The rules for forming *weight_name* are the same as those for forming *package_name* (page 222). *weight_name* must *exactly match* the corresponding *CAPACITY_NAME*.

weight_value is an unsigned floating-point value between 0 and 1000000 with at most three digits after the decimal point.

You can use these parameters to override the cluster-wide default package weight that corresponds to a given node capacity. You can define that cluster-wide default package weight by means of the *WEIGHT_NAME* and *WEIGHT_DEFAULT* parameters in the cluster configuration file (explicit default). If you do not define an explicit default (that is, if you define a *CAPACITY_NAME* in the cluster configuration file with no corresponding *WEIGHT_NAME* and *WEIGHT_DEFAULT*), the default weight is assumed to be zero (implicit default). Configuring *weight_name* and *weight_value* here in the package configuration file overrides the cluster-wide default (implicit or explicit), and assigns a particular weight to this package.

For more information, see "About Package Weights" (page 135). See also the discussion of the relevant parameters under "Cluster Configuration Parameters " (page 105), in the `cmmakepkg (1m)` and `cmquerycl (1m)` manpages, and in the cluster configuration and package configuration template files.

New for 11.19.

local_lan_failover_allowed

Specifies whether or not Serviceguard can transfer the package IP address to a standby LAN card in the event of a LAN card failure on a cluster node.

Legal values are `yes` and `no`. Default is `yes`.

monitored_subnet

A LAN subnet that is to be monitored for this package. Replaces legacy *SUBNET* which is still supported in the package configuration file for legacy packages; see "Configuring a Legacy Package" (page 289).

You can specify multiple subnets; use a separate line for each.

If you specify a subnet as a *monitored_subnet* the package will not run on any node not reachable via that subnet. This normally means that if the subnet is not up, the package will not run. (For cross-subnet configurations, in which a subnet may be configured on some nodes and not on others, see *monitored_subnet_access* below, *ip_subnet_node* (page 231), and "About Cross-Subnet Failover" (page 145).)

Typically you would monitor the *ip_subnet*, specifying it here as well as in the *ip_subnet* parameter (see below), but you may want to monitor other subnets as well; you can monitor any subnet that is configured into the cluster (via the *STATIONARY_IP* or *HEARTBEAT_IP* parameter in the cluster configuration file).

If any *monitored_subnet* fails, Serviceguard will switch the package to any other node specified by *node_name* which can communicate on the *monitored_subnet*s defined for this package. See the comments in the configuration file for more information and examples.

monitored_subnet_access

In cross-subnet configurations, specifies whether each *monitored_subnet* is accessible on all nodes in the package's node list (see *node_name* (page 223)), or only some. Valid values are PARTIAL, meaning that at least one of the nodes has access to the subnet, but not all; and FULL, meaning that all nodes have access to the subnet. The default is FULL, and it is in effect if *monitored_subnet_access* is not specified.

See also *ip_subnet_node* (page 231) and "About Cross-Subnet Failover" (page 145).

New for modular packages. For legacy packages, see "Configuring Cross-Subnet Failover" (page 295).

cluster_interconnect_subnet

Can be configured only for a multi-node package in a Serviceguard Extension for Real Application Cluster (SGeRAC) installation.

NOTE: Serviceguard supports both IPv4 and IPv6 for an *cluster_interconnect_subnet*, but before configuring an IPv6 subnet, make sure that your version of Oracle RAC also supports IPv6 for this subnet.

See the latest version of *Using Serviceguard Extension for RAC* at http://www.hp.com/go/hpux-serviceguard-docs for more information, including the status of Oracle RAC support for an IPv6 *cluster_interconnect_subnet*.

New for A.11.18 (for both modular and legacy packages).

ip_subnet

Specifies an IP subnet used by the package for relocatable addresses; see also *ip_address* (page 232) and "Stationary and Relocatable IP Addresses " (page 64). Replaces *SUBNET*, which is still supported in the package control script for legacy packages.

CAUTION: HP recommends that this subnet be configured into the cluster. You do this in the cluster configuration file by specifying a *HEARTBEAT_IP* or *STATIONARY_IP* under a *NETWORK_INTERFACE* on the same subnet, for each node in this package's *NODE_NAME* list. For example, an entry such as the following in the cluster configuration file configures subnet `192.10.25.0` (`lan1`) on node `ftsys9`:

```
NODE_NAME ftsys9
 NETWORK_INTERFACE lan1
  HEARTBEAT_IP 192.10.25.18
```

See "Cluster Configuration Parameters " (page 105) for more information.

If the subnet is not configured into the cluster, Serviceguard cannot manage or monitor it, and in fact cannot guarantee that it is available on all nodes in the package's *node-name* list (page 223) . Such a subnet is referred to as an external subnet, and relocatable addresses on that subnet are known as external addresses. If you use an external subnet, you risk the following consequences:

- A failed interface on this subnet will not fail over to a local standby. See "Local Switching " (page 66) for more information about this type of failover.
- If the subnet fails, the package will not fail over to an alternate node.
- Even if the subnet remains intact, if the package needs to fail over because of some other type of failure, it could fail to start on an adoptive node because the subnet is not available on that node.

For these reasons, configure all *ip_subnet*s into the cluster, unless you are using a networking technology, such as HyperFabric, that does not support DLPI. In such cases, follow instructions in the networking product's documentation to integrate the product with Serviceguard.

For each subnet used, specify the subnet address on one line and, on the following lines, the relocatable IP addresses that the package uses on that subnet. These will be configured when the package starts and unconfigured when it halts.

For example, if this package uses subnet `192.10.25.0` and the relocatable IP addresses `192.10.25.12` and `192.10.25.13`, enter:

```
ip_subnet 192.10.25.0
ip_address 192.10.25.12
ip_address 192.10.25.13
```

If you want the subnet to be monitored, specify it in the *monitored_subnet* parameter as well.

In a cross-subnet configuration, you also need to specify which nodes the subnet is configured on; see *ip_subnet_node* below. See also *monitored_subnet_access* (page 230) and "About Cross-Subnet Failover" (page 145).

This parameter can be set for failover packages only.

Can be added or deleted while the package is running.

ip_subnet_node

In a cross-subnet configuration, specifies which nodes an *ip_subnet* is configured on. If no *ip_subnet_node* are listed under an *ip_subnet*, it is assumed to be configured on all nodes in this package's *node_name* list (page 223).

Can be added or deleted while the package is running, with these restrictions:

- The package must not be running on the node that is being added or deleted.
- The node must not be the first to be added to, or the last deleted from, the list of *ip_subnet_node*s for this *ip_subnet*.

See also *monitored_subnet_access* (page 230) and "About Cross-Subnet Failover" (page 145).

New for modular packages. For legacy packages, see "Configuring Cross-Subnet Failover" (page 295).

ip_address

A relocatable IP address on a specified *ip_subnet* (page 231). Replaces *IP*, which is still supported in the package control script for legacy packages; see "Configuring a Legacy Package" (page 289).

For more information about relocatable IP addresses, see "Stationary and Relocatable IP Addresses " (page 64).

This parameter can be set for failover packages only.

Can be added or deleted while the package is running.

CAUTION: HP strongly recommends that you add relocatable addresses to packages *only* by editing *ip_address* (or *IP []* entries in the control script of a legacy package) and running `cmapplyconf (1m)`.

service_name

A service is a program or function which Serviceguard monitors as long as the package is up. *service_name* identifies this function and is used by the `cmrunserv` and `cmhaltserv` commands. You can configure a maximum of 30 services per package and 900 services per cluster.

The length and formal restrictions for the name are the same as for *package_name* (page 222). *service_name* must be unique among all packages in the cluster.

IMPORTANT: Restrictions on service names in previous Serviceguard releases were less stringent. Packages that specify services whose names do not conform to the above rules will continue to run, but if you reconfigure them, you will need to change the name; `cmcheckconf` and `cmapplyconf` will enforce the new rules.

Each service is defined by five parameters: *service_name*, *service_cmd*, *service_restart*, *service_fail_fast_enabled*, and *service_halt_timeout*. See the descriptions that follow.

The following is an example of fully defined service:

```
service_name                patricks-package4-ping
service_cmd                 "/usr/sbin/ping hasupt22"
service_restart             unlimited
service_fail_fast_enabled   no
service_halt_timeout        300
```

See the package configuration file for more examples.

For legacy packages, this parameter is in the package control script as well as the package configuration file.

service_cmd

The command that runs the application or service for this *service_name*, for example,

```
/usr/bin/X11/xclock -display 15.244.58.208:0
```

An absolute pathname is required; neither the *PATH* variable nor any other environment variable is passed to the command. The default shell is `/usr/bin/sh`.

NOTE: Be careful when defining service run commands. Each run command is executed in the following way:

- The `cmrunserv` command executes the run command.
- Serviceguard monitors the process ID (PID) of the process the run command creates.
- When the command 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*.

Make sure that each run command is the name of an actual service and that its process remains alive until the actual service stops. One way to manage this is to configure a package such that the service is actually a monitoring program that checks the health of the application that constitutes the main function of the package, and exits if it finds the application has failed. The application itself can be started by an *external_script* (page 239).

This parameter is in the package control script for legacy packages.

service_restart

The number of times Serviceguard will attempt to re-run the *service_cmd*. Valid values are `unlimited`, `none` or any positive integer value. Default is `none`.

If the value is `unlimited`, the service will be restarted an infinite number of times. If the value is `none`, the service will not be restarted.

This parameter is in the package control script for legacy packages.

service_fail_fast_enabled

Specifies whether or not Serviceguard will halt the node (system reset) on which the package is running if the service identified by *service_name* fails. Valid values are `yes` and `no`. Default is `no`, meaning that failure of this service will not cause the node to halt. `yes` is not meaningful if *service_restart* is set to `unlimited`.

service_halt_timeout

The length of time, in seconds, Serviceguard will wait for the service to halt before forcing termination of the service's process. The maximum value is 4294.

The value should be large enough to allow any cleanup required by the service to complete.

If no value is specified, a zero timeout will be assumed, meaning that Serviceguard will not wait any time before terminating the process.

resource_name

The name of a resource to be monitored.

resource_name, in conjunction with *resource_polling_interval*, *resource_start* and *resource_up_value*, defines an Event Monitoring Service (EMS) dependency.

In legacy packages, *RESOURCE_NAME* in the package configuration file requires a corresponding *DEFERRED_RESOURCE_NAME* in the package control script.

You can find a list of resources in Serviceguard Manager (`Configuration -> Create Package -> Monitored Resources -> Available EMS resources`), or in the documentation supplied with the resource monitor.

A maximum of 60 EMS resources can be defined per cluster. Note also the limit on *resource_up_value* (see below).

The maximum length of the *resource_name* string is 1024 characters.

See "Parameters for Configuring EMS Resources" (page 128) for more information and example.

resource_polling_interval

How often, in seconds, the resource identified by *resource_name* (page 233) is to be monitored. Default is 60 seconds. The minimum value is 1. (There is no practical maximum.)

resource_start

Specifies when Serviceguard will begin monitoring the resource identified by *resource_name*. Valid values are `automatic` and `deferred`.

`automatic` means Serviceguard will begin monitoring the resource as soon as the node joins the cluster.`deferred` means Serviceguard will begin monitoring the resource when the package starts.

resource_up_value

Defines a criterion used to determine whether the resource identified by *resource_name* is up.

Requires an operator and a value. Values can be string or numeric. The legal operators are `=`, `!=`, `>`, `<`, `>=`, or`<=`, depending on the type of value. If the type is string, then only `=` and `!=` are valid. If the string contains white space, it must be enclosed in quotes. String values are case-sensitive.

The maximum length of the entire *resource_up_value* string is 1024 characters.

You can configure a total of 15 *resource_up_value*s per package. For example, if there is only one resource (*resource_name*) in the package, then a maximum of 15 *resource_up_value*s can be defined. If two *resource_name*s are defined and one of them has 10 *resource_up_value*s, then the other *resource_name* can have only 5 *resource_up_value*s.

concurrent_vgchange_operations

Specifies the number of concurrent volume group activations or deactivations allowed during package startup or shutdown.

Legal value is any number greater than zero. The default is `1`.

If a package activates a large number of volume groups, you can improve the package's start-up and shutdown performance by carefully tuning this parameter. Tune performance by increasing the number a little at a time and monitoring the effect on performance at each step; stop increasing it, or reset it to a lower level, as soon as performance starts to level off or decline. Factors you need to take into account include the number of CPUs, the amount of available memory, the HP-UX kernel settings for *nfile* and *nproc*, and the number and characteristics of other packages that will be running on the node.

NOTE: If you *set concurrent_vgchange_operations* to a value greater than 1, you may see messages such as this in the package log file:

```
Cannot lock "/etc/lvmconf//lvm_lock" still trying..."
```

This is an informational message that can be safely ignored.

---

enable_threaded_vgchange

Indicates whether multi-threaded activation of volume groups (`vgchange -T`) is enabled. New for modular packages. Available on HP-UX 11i v3 only.

Legal values are zero (disabled) or `1` (enabled). The default is zero.

Set *enable_threaded_vgchange* to `1` to enable `vgchange -T` for all of a package's volume groups. This means that when each volume group is activated, physical volumes (disks or LUNs) are attached to the volume group in parallel, and mirror copies of logical volumes are synchronized

in parallel, rather than serially. That can improve a package's startup performance if its volume groups contain a large number of physical volumes.

Note that, in the context of a Serviceguard package, this affects the way physical volumes are activated *within* a volume group; *concurrent_vgchange_operations* (page 234) controls how many volume groups the package can activate simultaneously.

IMPORTANT: Make sure you read the configuration file comments for both *concurrent_vgchange_operations* and *enable_threaded_vgchange* before configuring these options, as well as the `vgchange (1m)` manpage.

vgchange_cmd

Specifies the method of activation for each HP-UX Logical Volume Manager (LVM) volume group identified by a `vg` entry (see (page 236)). Replaces *VGCHANGE* which is still supported in the package control script for legacy packages; see "Configuring a Legacy Package" (page 289).

The default is `vgchange -a e`.

The configuration file contains several other `vgchange` command variants; either uncomment one of these and comment out the default, or use the default. For more information, see the explanations in the configuration file, "LVM Planning " (page 96), and "Creating the Storage Infrastructure and file systems with LVM, VxVM and CVM" (page 168).

IMPORTANT: Volume groups for multi-node and system multi-node packages must be activated in shared mode: `vgchange -a s`, which is only available if the add-on product Serviceguard Extension for Real Application Cluster (SGeRAC) is installed. (See the latest version of *Using Serviceguard Extension for RAC* at http://www.hp.com/go/hpux-serviceguard-docs for more information.) Shared LVM volume groups must not contain a file system.

(For more information about LVM, see the *Logical Volume Management* volume of the *HP-UX System Administrator's Guide* under `HP-UX 11i v3` at http://www.hp.com/go/hpux-core-docs.)

NOTE: A given package can use LVM volume groups, VxVM volume groups, CVM volume groups, or any combination.

cvm_activation_cmd

Specifies the method of activation for Veritas Cluster Volume Manager (CVM) disk groups. The default is

```
vxdg -g \${DiskGroup} set activation=readonly
```

The configuration file contains several other `vxdg` command variants; either uncomment one of these and comment out the default, or use the default. For more information, see the explanations in the configuration file, and the sections "CVM and VxVM Planning " (page 98) and "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208).

vxvol_cmd

Specifies the method of recovery for mirrored VxVM volumes. Replaces *VXVOL*, which is still supported in the package control script for legacy packages; see "Configuring a Legacy Package" (page 289).

If recovery is found to be necessary during package startup, by default the script will pause until the recovery is complete. To change this behavior, comment out the line

```
vxvol_cmd "vxvol -g \${DiskGroup} startall"
```

in the configuration file, and uncomment the line

```
vxvol_cmd "vxvol -g \${DiskGroup} -o bg startall"
```

This allows package startup to continue while mirror re-synchronization is in progress.

vg

Specifies an LVM volume group (one per *vg*, each on a new line) on which a file system needs to be mounted. A corresponding *vgchange_cmd* (page 235) specifies how the volume group is to be activated. The package script generates the necessary file system commands on the basis of the *fs_* parameters (page 238).

NOTE: A volume group must be configured into the cluster (via the *VOLUME_GROUP* parameter) before you can configure it into a modular package. For more information about cluster parameters, see "Cluster Configuration Parameters " (page 105).

cvm_dg

Specifies a CVM disk group (one per *cvm_dg*, each on a new line) used by this package. CVM disk groups must be specified whether file systems need to be mounted on them or not. A corresponding *cvm_activation_cmd* (page 235) specifies how the disk group is to be activated.

Any package using a CVM disk group must declare a dependency (see the entries for *dependency_name* and related parameters starting on (page 227)) on the CVM system multi-node package. See "Preparing the Cluster for Use with CVM " (page 209).

vxvm_dg

Specifies a VxVM disk group (one per *vxvm_dg*, each on a new line) on which a file system needs to be mounted. See the comments in the package configuration file, and "Creating the Storage Infrastructure and file systems with LVM, VxVM and CVM" (page 168), for more information.

vxvm_dg_retry

Specifies whether to retry the import of a VxVM disk group, using `vxdisk scandisks` to check for any missing disks that might have caused the import to fail. Equivalent to *VXVM_DG_RETRY* in the legacy package control script.

Legal values are `yes` and `no`. `yes` means `vxdisk scandisks` will be run in the event of an import failure. The default is `no`.

HP recommends setting this parameter to `yes` in Metrocluster installations using EMC SRDF.

IMPORTANT: `vxdisk scandisks` can take a long time in the case of a large IO subsystem.

deactivation_retry_count

Specifies how many times the package shutdown script will repeat an attempt to deactivate a volume group (LVM) or disk group (VxVM, CVM) before failing.

Legal value is zero or any greater number. As of A.11.18, the default is 2.

kill_processes_accessing_raw_devices

Specifies whether or not to kill processes that are using raw devices (for example, database applications) when the package shuts down. Default is `no`. See the comments in the package configuration file for more information.

File system parameters

A package can activate one or more storage groups on startup, and mount logical volumes to file systems. At halt time, the package 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 local file system you specify in the package configuration file (*fs_name* (page 238)), you must identify a logical volume, the mount point, the `mount`, `umount`, and `fsck` options and type of the file system; for example:

```
fs_name /dev/vg01/lvol1
fs_directory /pkg01aa
fs_mount_opt "-o rw"
fs_umount_opt "-s"
fs_fsck_opt "-s"
fs_type "vxfs"
```

A logical volume can be created in an LVM volume group, a Veritas VxVM disk group, or a Veritas CVM disk group. Logical volumes can be entered in any order, regardless of the type of storage group. The parameter explanations that follow provide more detail.

For an NFS-imported file system, see the discussion under *fs_name* (page 238) and *fs_server* (page 238).

concurrent_fsck_operations

The number of concurrent `fsck` operations allowed on file systems being mounted during package startup.

Legal value is any number greater than zero. The default is 1.

If the package needs to run `fsck` on a large number of file systems, you can improve performance by carefully tuning this parameter during testing (increase it a little at time and monitor performance each time).

concurrent_mount_and_umount_operations

The number of concurrent `mount`s and `umount`s to allow during package startup or shutdown.

Legal value is any number greater than zero. The default is 1.

If the package needs to mount and unmount a large number of file systems, you can improve performance by carefully tuning this parameter during testing (increase it a little at time and monitor performance each time).

fs_mount_retry_count

The number of `mount` retries for each file system. Legal value is zero or any greater number. The default is zero.

If the mount point is busy at package startup and *fs_mount_retry_count* is set to zero, package startup will fail.

If the mount point is busy and *fs_mount_retry_count* is greater than zero, the startup script will attempt to kill the user process responsible for the busy mount point (`fuser -ku`) and then try to mount the file system again. It will do this the number of times specified by *fs_mount_retry_count*.

If the mount still fails after the number of attempts specified by *fs_mount_retry_count*, package startup will fail.

This parameter is in the package control script for legacy packages. See "Configuring a Legacy Package" (page 289).

fs_umount_retry_count

The number of `umount` retries for each file system. Replaces *FS_UMOUNT_COUNT*, which is still supported in the package control script for legacy packages; see "Configuring a Legacy Package" (page 289).

Legal value is any greater number greater than zero. The default is `1`. Operates in the same way as *fs_mount_retry_count* (page 237).

fs_name

- For a local file system, this parameter, in conjunction with *fs_directory*, *fs_type*, *fs_mount_opt*, *fs_umount_opt*, and *fs_fsck_opt*, specifies the file system that is to be mounted by the package. *fs_name* must specify the block devicefile for a logical volume.
- For an NFS-imported file system, the additional parameters required are *fs_server*, *fs_directory*, *fs_type*, and *fs_mount_opt*; see *fs_server* (page 238) for an example.

△ **CAUTION:** Before configuring an NFS-imported file system into a package, make sure you have read and understood the rules and guidelines under "Planning for NFS-mounted File Systems" (page 125), and configured the cluster parameter *CONFIGURED_IO_TIMEOUT_EXTENSION*, described under "Cluster Configuration Parameters " (page 105).

Replaces *LV*, which is still supported in the package control script for legacy packages.

file systems are mounted in the order specified in this file, and unmounted in the reverse order.

NOTE: A volume group must be defined in this file, using *vg* (page 236), for each logical volume specified by an *fs_name* entry.

See "File system parameters" (page 236), and the comments in the FILESYSTEMS section of the configuration file, for more information and examples. See also "Logical Volume and File System Planning " (page 120), and the mount (1m) and mount_nfs (1m) manpages.

fs_server

The name or IP address (IPv4 or IPv6) of the NFS server for an NFS-imported file system. In this case, you must also set *fs_type* to nfs, and set *fs_mount_opt* to "-o llock" (in addition to any other values you use for *fs_mount_opt*). *fs_name* specifies the directory to be imported from *fs_server*, and *fs_directory* specifies the local mount point.

For example:

```
fs_name                 /var/opt/nfs/share1
fs_server               wagon
fs_directory            /nfs/mnt/share1
fs_type                 nfs
fs_mount_opt            "-o llock"
#fs_umount_opt
#fs_fsck_opt
```

Note that *fs_umount_opt* is optional and *fs_fsck_opt* is not used for an NFS-imported file system. (Both are left commented out in this example.)

fs_directory

The root of the file system specified by *fs_name*. Replaces *FS*, which is still supported in the package control script for legacy packages; see "Configuring a Legacy Package" (page 289). See the mount (1m) manpage for more information.

fs_type

The type of the file system specified by *fs_name*. This parameter is in the package control script for legacy packages.

For an NFS-imported file system, this must be set to nfs. See the example under "*fs_server*" (page 238).

See the mount (1m) and fstyp (1m) manpages for more information.

fs_mount_opt

The mount options for the file system specified by *fs_name*. This parameter is in the package control script for legacy packages.

See the mount (1m) manpage for more information.

fs_umount_opt

The umount options for the file system specified by *fs_name*. This parameter is in the package control script for legacy packages.

Using the -s option of umount will improve startup performance if the package uses a large number of file systems. See the mount (1m) manpage for more information.

fs_fsck_opt

The fsck options for the file system specified by *fs_name*. Using the -s (safe performance mode) option of fsck will improve startup performance if the package uses a large number of file systems. This parameter is in the package control script for legacy packages.

See the fsck (1m) manpage for more information.

*pev_*

Specifies a package environment variable that can be passed to an *external_pre_script*, *external_script*, or both, by means of the cmgetpkgenv (1m) command. New for modular packages.

The variable name must be in the form pev_*<variable_name>* and contain only alphanumeric characters and underscores. The letters pev (upper-case or lower-case) followed by the underscore (_) are required.

The variable name and value can each consist of a maximum of *MAXPATHLEN* characters (1024 on HP-UX systems).

You can define more than one variable. See "About External Scripts" (page 142), as well as the comments in the configuration file, for more information.

external_pre_script

The full pathname of an external script to be executed before volume groups and disk groups are activated during package startup, and after they have been deactivated during package shutdown (that is, effectively the first step in package startup and last step in package shutdown). New for modular packages.

If more than one *external_pre_script* is specified, the scripts will be executed on package startup in the order they are entered into this file, and in the reverse order during package shutdown.

See "About External Scripts" (page 142), and the comments in the configuration file, for more information and examples.

external_script

The full pathname of an external script. This script is often the means of launching and halting the application that constitutes the main function of the package. New for modular packages.

The script is executed on package startup after volume groups and file systems are activated and IP addresses are assigned, but before services are started; and during package shutdown after services are halted but before IP addresses are removed and volume groups and file systems deactivated.

If more than one *external_script* is specified, the scripts will be executed on package startup in the order they are entered into this file, and in the reverse order during package shutdown.

See "About External Scripts" (page 142), and the comments in the configuration file, for more information and examples.

user_name

Specifies the name of a user who has permission to administer this package. See also *user_host* and *user_role*; these three parameters together define the access-control policy for this package (see "Controlling Access to the Cluster" (page 183)). These parameters must be defined in this order: *user_name*, *user_host*, *user_role*.

Legal values for *user_name* are any_user or a maximum of eight login names from /etc/passwd on *user_host*.

NOTE: Be careful to spell any_user exactly as given; otherwise Serviceguard will interpret it as a user name.

Note that the only *user_role* that can be granted in the package configuration file is package_admin for this particular package; you grant other roles in the cluster configuration file. See "Setting up Access-Control Policies" (page 185) for further discussion and examples.

user_host

The system from which a user specified by *user_name* can execute package-administration commands.

Legal values are any_serviceguard_node, or cluster_member_node, or a specific cluster node. If you specify a specific node it must be the official hostname (the *hostname* portion, and *only* the*hostname* portion, of the fully qualified domain name). As with *user_name*, be careful to spell the keywords exactly as given.

user_role

Must be package_admin, allowing the user access to the cmrunpkg, cmhaltpkg, and cmmodpkg commands (and the equivalent functions in Serviceguard Manager) for this package, and to the Monitor role for the cluster. See "Controlling Access to the Cluster" (page 183) for more information.

Additional Parameters Used Only by Legacy Packages

IMPORTANT: The following parameters are used only by legacy packages. Do not try to use them in modular packages. See "Configuring a Legacy Package" (page 289) for more information.

PATH	Specifies the path to be used by the script.
SUBNET	Specifies the IP subnets that are to be monitored for the package.
RUN_SCRIPT and *HALT_SCRIPT*	Use the full pathname of each script. These two parameters allow you to separate package run instructions and package halt instructions for legacy packages into separate scripts if you need to. In this case, make sure you include identical configuration information (such as node names, IP addresses, etc.) in both scripts. In most cases, though, HP recommends that you use the same script for both run and halt instructions. (When the package starts, the script is passed the parameter *start*; when it halts, it is passed the parameter *stop*.)
DEFERRED_RESOURCE_NAME	Add *DEFERRED_RESOURCE_NAME* to a legacy package control script for any resource that has a *RESOURCE_START* setting of DEFERRED.

Generating the Package Configuration File

When you have chosen the configuration modules your package needs (see "Choosing Package Modules" (page 217)), you are ready to generate a package configuration file that contains those modules. This file will consist of a base module (usually failover, multi-node or system multi-node) plus the modules that contain the additional parameters you have decided to include.

Before You Start

Before you start building a package, create a subdirectory for it in the `$SGCONF` directory (`/etc/cmcluster`); for example:

```
mkdir $SGCONF/pkg1
```

cmmakepkg Examples

The `cmmakepkg` command generates a package configuration file. Some examples follow; see the `cmmakepkg (1m)` manpage for complete information. All the examples create an editable configuration file `pkg1.conf` in the `$SGCONF/pkg1` directory.

NOTE: If you do not include a base module (or `default` or `all`) on the `cmmakepkg` command line, `cmmakepkg` will ignore the modules you specify and generate a default configuration file containing all the parameters.

For a complex package, or if you are not yet sure which parameters you will need to set, the default may be the best choice; see the first example below.

You can use the `-v` option with `cmmakepkg` to control how much information is displayed online or included in the configuration file. Valid values are `0`, `1` and `2`. `-v 0` removes all comments; `-v 1` includes a brief heading for each parameter; `-v 2` provides a full description of each parameter. The default is level 2.

- To generate a configuration file that contains all the optional modules:

  ```
  cmmakepkg $SGCONF/pkg1/pkg1.conf
  ```

- To create a generic failover package (that could be applied with out editing):

  ```
  cmmakepkg -n pkg1 -m sg/failover $SGCONF/pkg1/pkg1.conf
  ```

- To generate a configuration file for a failover package that uses relocatable IP addresses and runs an application that requires file systems to be mounted at run time (enter the command all on one line):

  ```
  cmmakepkg -m sg/failover -m sg/package_ip -m sg/service -m
  sg/filesystem -m sg/volume_group $SGCONF/pkg1/pkg1.conf
  ```

- To generate a configuration file for a multi-node package that monitors cluster resources (enter the command all on one line):

  ```
  cmmakepkg -m sg/multi_node -m sg/resource $SGCONF/pkg1/pkg1.conf
  ```

- To generate a configuration file for a failover package that runs an application that requires another package to be up (enter the command all on one line):

  ```
  cmmakepkg -m sg/failover -m sg/dependency -m sg/service
  $SGCONF/pkg1/pkg1.conf
  ```

- To generate a configuration file adding the `services` module to an existing package (enter the command all on one line):

  ```
  cmmakepkg -i $SGCONF/pkg1/pkg1.conf -m sg/service
  $SGCONF/pkg1/pkg1_v2.conf
  ```

NOTE: You can add more than one module at a time.

Next Step

The next step is to edit the configuration file you have generated; see "Editing the Configuration File" (page 242).

Editing the Configuration File

When you have generated the configuration file that contains the modules your package needs (see "Generating the Package Configuration File" (page 241)), you need to edit the file to set the package parameters to the values that will make the package function as you intend.

IMPORTANT: Do not edit the package configuration file of a Veritas Cluster Volume Manager (CVM) or Cluster File System (CFS) multi-node or system multi-node package.

Create `SG-CFS-pkg` by means of the `cmapplyconf` command. Create and modify `SG-CFS-DG-id#` and `SG-CFS-MP-id#` using `cfs*` commands.

It is a good idea to configure complex failover packages in stages, as follows:

1. Configure volume groups and mount points only.
2. Check and apply the configuration; see "Verifying and Applying the Package Configuration" (page 245).
3. Run the package and ensure that it can be moved from node to node.

   NOTE: `cmcheckconf` and `cmapplyconf` check for missing mount points, volume groups, etc.

4. Halt the package.
5. Configure package IP addresses and application services.
6. Run the package and ensure that applications run as expected and that the package fails over correctly when services are disrupted. See "Testing the Package Manager " (page 308).

Use the following bullet points as a checklist, referring to the "Package Parameter Explanations" (page 222), and the comments in the configuration file itself, for detailed specifications for each parameter.

NOTE: Optional parameters are commented out in the configuration file (with a # at the beginning of the line). In some cases these parameters have default values that will take effect unless you uncomment the parameter (remove the #) and enter a valid value different from the default. Read the surrounding comments in the file, and the explanations in this chapter, to make sure you understand the implications both of accepting and of changing a given default.

In all cases, be careful to uncomment each parameter you intend to use and assign it the value you want it to have.

- *`package_name`*. Enter a unique name for this package. Note that there are stricter formal requirements for the name as of A.11.18.
- *`package_type`*. Enter `failover`, `multi_node`, or `system_multi_node`. (`system_multi_node` is reserved for special-purpose packages supplied by HP.) Note that there are restrictions if another package depends on this package; see "About Package Dependencies" (page 128).

  See "Types of Package: Failover, Multi-Node, System Multi-Node" (page 217) for more information.
- *`node_name`*. Enter the name of each cluster node on which this package can run, with a separate entry on a separate line for each node.

- *auto_run*. For failover packages, enter yes to allow Serviceguard to start the package on the first available node specified by *node_name*, and to automatically restart it later if it fails. Enter no to keep Serviceguard from automatically starting the package.
- *node_fail_fast_enabled*. Enter yes to cause the node to be halted (system reset) if the package fails; otherwise enter no.

  For system multi-node packages, you must enter yes.
- *run_script_timeout* and *halt_script_timeout*. Enter the number of seconds Serviceguard should wait for package startup and shutdown, respectively, to complete; or leave the default, no_timeout; see (page 224).
- *successor_halt_timeout*. Used if other packages depend on this package; see "About Package Dependencies" (page 128).
- *script_log_file* (page 225).
- *log_level* (page 226).
- *failover_policy* (page 226). Enter configured_node or min_package_node.

  (This parameter can be set for failover packages only.)
- *failback_policy* (page 227) Enter automatic or manual.

  (This parameter can be set for failover packages only.)
- If this package will depend on another package or packages, enter values for *dependency_name*, (page 227)*dependency_condition*, *dependency_location*, and optionally *priority*.

  See "About Package Dependencies" (page 128) for more information.

  NOTE: The package(s) this package depends on must already be part of the cluster configuration by the time you validate this package (via cmcheckconf(1m); see "Verifying and Applying the Package Configuration" (page 245)); otherwise validation will fail.
- To configure package weights, use the *weight_name* and *weight_value* parameters (page 229). See "About Package Weights" (page 135) for more information.
- *local_lan_failover_allowed*. Enter yes to permit switching of the package IP address to a standby LAN card on the local node, or no to keep the package address from switching locally.

  For multi-node packages, you must enter no.
- Use the *monitored_subnet* parameter to specify a subnet that is to be monitored for this package. If there are multiple subnets, repeat the parameter as many times as needed, on a new line each time.

  In a cross-subnet configuration, configure the additional *monitored_subnet_access* parameter for each *monitored_subnet* as necessary; see "About Cross-Subnet Failover" (page 145) for more information.
- If this is a Serviceguard Extension for Oracle RAC (SGeRAC) installation, you can use the *cluster_interconnect_subnet* parameter(page 230).
- If your package will use relocatable IP addresses, enter the *ip_subnet* and *ip_address* addresses. See the parameter descriptions (page 231) for rules and restrictions.

  In a cross-subnet configuration, configure the additional *ip_subnet_node* parameter for each *ip_subnet* as necessary; see "About Cross-Subnet Failover" (page 145) for more information.

- For each service the package will run, enter values for the following parameters (page 232):
  - *service_name* (for example, a daemon or long-running process)
  - *service_cmd* (for example, the command that starts the process)
  - *service_fail_fast_enabled* and *service_halt_timeout* if you need to change them from their defaults.
  - *service_restart* if you want the package to restart the service if it exits. (A value of unlimited can be useful if you want the service to execute in a loop, rather than exit and halt the package.)
- To configure the package to monitor a registered EMS resource, enter values for the following parameters (page 233):
  - *resource_name*
  - *resource_polling_interval*
  - *resource_up_value*
  - *resource_start*

  See "Parameters for Configuring EMS Resources" (page 128) for more information and an example.
- If the package needs to mount LVM volumes to file systems, use the *vg* parameters (page 236) to specify the names of the volume groups to be activated, select the appropriate *vgchange_cmd*, and use the *fs_* options in the FILESYSTEMS portion of the configuration file to specify the options for mounting and unmounting the file systems. Do not use the *vxvm_dg* or *cvm_dg* parameters for LVM volume groups. Enter each volume group on a separate line, for example:

  ```
  vg vg01
  vg vg02
  ```
- If you are using CVM disk group for raw storage (without CFS), use *cvm_dg* parameters (page 236) to specify the names of the disk groups to be activated, select the appropriate *cvm_activation_cmd*, and create a dependency (page 227) on SG_CFS_pkg. See "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208).

  Do not use the *vxvm_dg* or *vg* parameters for CVM disk groups.

  Do not include CFS-based disk groups in the package configuration file; they are activated by the CFS multi-node packages before standard packages are started. See "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190).
- If you are using VxVM disk groups without CVM, enter the names of VxVM disk groups that will be imported using *vxvm_dg* parameters. See "How Control Scripts Manage VxVM Disk Groups" (page 246).
- If you are using mirrored VxVM disks, use (page 235) to specify the mirror recovery option to be used by vxvol.
- Specify the file system mount retry and unmount count options (see (page 237)).
- You can specify a *deactivation_retry_count* for LVM, CVM, and VxVM volume groups. See (page 236).
- You can specify whether or not to kill processes activating raw devices on shutdown; see (page 236).

- If your package uses a large number of volume groups or disk groups, or mounts a large number of file systems, consider increasing the values of the following parameters:
  - *concurrent_vgchange_operations* (page 234)
  - *concurrent_fsck_operations* (page 237)
  - *concurrent_mount_and_umount_operations* (page 237)

  You can also use the *fsck_opt* and *fs_umount_opt* parameters (page 239) to specify the `-s` option of the `fsck` and `mount/umount` commands.
- You can use the *pev_* parameter (page 239) to specify a variable to be passed to external scripts. Make sure the variable name begins with the upper-case or lower-case letters `pev` and an underscore (_). You can specify more than one variable. See "About External Scripts" (page 142), and the comments in the configuration file, for more details.
- If you want the package to run an external "pre-script" during startup and shutdown, use the *external_pre_script* parameter (see (page 239)) to specify the full pathname of the script, for example `/etc/cmcluster/pkg1/pre_script1`.
- If the package will run an external script, use the *external_script* parameter (see (page 239)) to specify the full pathname of the script, for example `/etc/cmcluster/pkg1/script1`.

  See "About External Scripts" (page 142), and the comments in the configuration file, for more information.
- 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.

  These files are in `/opt/cmcluster/toolkit/DB/`. Separate toolkits are available for Oracle, Informix, and Sybase.

  In addition to the standard package script, you use the special script that is provided for the database. To set up these scripts, follow the instructions in the README file provided with each toolkit.
- Configure the Access Control Policy for up to eight specific users or `any_user`.

  The only user role you can configure in the package configuration file is `package_admin` for the package in question. Cluster-wide roles are defined in the cluster configuration file. See "Setting up Access-Control Policies" (page 185) for more information.

Verifying and Applying the Package Configuration

Serviceguard checks the configuration you enter and reports any errors.

Use a command such as the following to verify the content of the package configuration file you have created, for example:

```
cmcheckconf -v -P $SGCONF/pkg1/pkg1.conf
```

Errors are displayed on the standard output. If necessary, re-edit the file to correct any errors, then run `cmcheckconf` again until it completes without errors.

The following items are checked:

- The package name is valid, and at least one *node_name* entry is included.
- There are no duplicate parameter entries (except as permitted for multiple volume groups, etc.)
- Values for all parameters are within permitted ranges.
- Configured resources are available on cluster nodes.

- File systems and volume groups are valid (for a modular package, `cmcheckconf` checks that volume groups are already configured into the cluster).
- Services are executable.
- Any package that this package depends on is already part of the cluster configuration.

For more information, see the manpage for `cmcheckconf (1m)` and "Checking Cluster Components" (page 261).

When `cmcheckconf` has completed without errors, apply the package configuration, for example:

```
cmapplyconf -P $SGCONF/pkg1/pkg1.conf
```

This adds the package configuration information to the binary cluster configuration file in the `$SGCONF` directory (normally `/etc/cmcluster`) and distributes it to all the cluster nodes.

NOTE: For modular packages, you now need to distribute any external scripts identified by the *external_pre_script* and *external_script* parameters.

But if you are accustomed to configuring legacy packages, note that you *do not* have to create a separate package control script for a modular package, or distribute it manually. (You do still have to do this for legacy packages; see "Configuring a Legacy Package" (page 289).)

Adding the Package to the Cluster

You can add the new package to the cluster while the cluster is running, subject to the value of *MAX_CONFIGURED_PACKAGES* in the cluster configuration file. See "Adding a Package to a Running Cluster" (page 299).

How Control Scripts Manage VxVM Disk Groups

VxVM disk groups (other than those managed by 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.

△ **CAUTION:** Although Serviceguard uses the `-C` option within the package control script framework, this option should not normally be used from the command line. Chapter 8: "Troubleshooting Your Cluster" (page 308), 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.

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

You can check status using Serviceguard Manager or from a cluster node's command line.

Reviewing Cluster and Package Status 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 means of the `cmviewcl` command: **`cmviewcl -v`**

NOTE: This section provides examples of the information you can obtain from `cmviewcl`; for complete details, see the manpages `cmviewcl(1m)` and `cmviewcl(5)`. Note that the `-f line` option provides information not available with other options.

You can use the `cmviewcl` command without root access; in clusters running Serviceguard version A.11.16 or later, grant access by assigning the Monitor role to the users in question. In earlier versions, allow access by adding <*nodename*> <*nonrootuser*> to the `cmclnodelist` file.

`cmviewcl -v` 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 to specify the release format you want, for example:

```
cmviewcl -r A.11.16
```

(See the `cmviewcl(1m)` manpage for the supported release formats.) The formatting options let you choose a style: the tabulated format is designed for viewing; the line format is designed for scripting, and is easily parsed.

See the manpage for a detailed description of other `cmviewcl` options.

Viewing Dependencies

The `cmviewcl -v` command output lists dependencies throughout the cluster. For a specific package's dependencies, use the `-p` <*pkgname*> option.

Viewing CFS Multi-Node Information

If you are running Veritas Cluster File System (CFS) cluster, you can use `cfs` commands to see multi-node package configuration information, status, and dependencies; for example:

```
cfsdgadm show_package <diskgroup>
cfsmntadm show_package <mountpoint>
getconf -p <mnpkg> | grep dependency
```

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, as shown by `cmviewcl`, can 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` (as an active 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`. Active members of the cluster will see a node in this state if that node was active in a 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`. Other nodes will see a node in this state after the node has gracefully left the active cluster, for instance as result of a `cmhaltnode` command.
- `Unknown`. 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 master control script is active.
- `down` — The package master control script is not active.
- `start_wait` — A `cmrunpkg` command is in progress for this package. The package is waiting for packages it depends on (predecessors) to start before it can start.
- `starting` — The package is starting. The package master control script is running.
- `halting` — A `cmhaltpkg` command is in progress for this package and the halt script is running.

- halt_wait — A cmhaltpkg command is in progress for this package. The package is waiting to be halted, but the halt script cannot start because the package is waiting for packages that depend on it (successors) to halt. The parameter description for *successor_halt_timeout* (page 225) provides more information.
- failing — The package is halting because it, or a package it depends on, has failed.
- fail_wait — The package is waiting to be halted because the package or a package it depends on has failed, but must wait for a package it depends on to halt before it can halt.
- relocate_wait — The package's halt script has completed or Serviceguard is still trying to place the package.
- reconfiguring — The node where this package is running is adjusting the package configuration to reflect the latest changes that have been applied.
- reconfigure_wait — The node where this package is running is waiting to adjust the package configuration to reflect the latest changes that have been applied.
- unknown — Serviceguard could not determine the status at the time cmviewcl was run.

A system multi-node package is up when it is running on *all* the active cluster nodes. A multi-node package is up if it is running on *any* of its configured nodes.

A system multi-node package can have a status of changing, meaning the package is in transition on one or more active nodes.

The state of the package can be one of the following:

- starting — The package is starting. The package master control script is running.
- start_wait — A cmrunpkg command is in progress for this package. The package is waiting for packages it depends on (predecessors) to start before it can start.
- running — Services are active and being monitored.
- halting — A cmhaltpkg command is in progress for this package and the halt script is running.
- halt_wait — A cmhaltpkg command is in progress for this package. The package is waiting to be halted, but the halt script cannot start because the package is waiting for packages that depend on it (successors) to halt. The parameter description for *successor_halt_timeout* (page 225) provides more information.
- halted — The package is down and halted.
- failing — The package is halting because it, or a package it depends on, has failed.
- fail_wait — The package is waiting to be halted because the package or a package it depends on has failed, but must wait for a package it depends on to halt before it can halt.
- failed — The package is down and failed.
- relocate_wait — The package's halt script has completed or Serviceguard is still trying to place the package.
- maintenance — The package is in maintenance mode; see "Maintaining a Package: Maintenance Mode" (page 273).
- reconfiguring — The node where this package is running is adjusting the package configuration to reflect the latest changes that have been applied.
- reconfigure_wait — The node where this package is running is waiting to adjust the package configuration to reflect the latest changes that have been applied.
- unknown — Serviceguard could not determine the state at the time cmviewcl was run.

The following states are possible only for multi-node packages:

- `blocked` — The package has never run on this node, either because a dependency has not been met, or because *`auto_run`* is set to no.
- `changing` — The package is in a transient state, different from the status shown, on some nodes. For example, a status of `starting` with a state of `changing` would mean that the package was starting on at least one node, but in some other, transitory condition (for example, `failing`) on at least one other node.

Package Switching Attributes

`cmviewcl` shows the following package switching information:

- `AUTO_RUN`: Can be `enabled` or `disabled`. For failover packages, `enabled` means that the package starts when the cluster starts, and Serviceguard can switch the package to another node in the event of failure.

  For system multi-node packages, `enabled` means an instance of the package can start on a new node joining the cluster (`disabled` means it will not).

- (Switching for a node.) Every failover package is marked `enabled` or `disabled` for each node that is either a primary or adoptive node for the package.

  For failover packages, `enabled` means that the package can switch to the specified node. `disabled` means that the package cannot switch to the specified node until the node is enabled to run the package via the `cmmodpkg` command.

  For multi-node packages, node switching `disabled` means the package cannot start on that node.

Service Status

Services have only status, as follows:

- `Up`. The service is being monitored.
- `Down`. The service is not running. It may not have started, or have halted or failed.
- `Unknown`.

Network Status

The network interfaces have only status, as follows:

- `Up`.
- `Down`.
- `Unknown`. Serviceguard cannot determine whether the interface is up or down. A standby interface has this status.

Failover and Failback Policies

Failover packages can be configured with one of two values for the *`failover_policy`* parameter (page 226), as displayed in the output of `cmviewcl -v`:

- `configured_node`. The package fails over to the next node in the *`node_name`* list in the package configuration file (page 223).
- `min_package_node`. The package fails over to the node in the cluster that has the fewest running packages.

Failover packages can also be configured with one of two values for the *failback_policy* parameter (page 227), and these are also displayed in the output of `cmviewcl -v`:

- `automatic`: Following a failover, a package returns to its primary node when the primary node becomes available again.
- `manual`: Following a failover, a package will run on the adoptive node until moved back to its original node by a system administrator.

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       (current)
      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      ftsys10

      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      (current)
      Alternate    up           enabled      ftsys9
```

NOTE: The `Script_Parameters` section of the `PACKAGE` output of `cmviewcl` shows the `Subnet` status only for the node that the package is running on. In a cross-subnet configuration, in which the package may be able to fail over to a node on another subnet, that other subnet is not shown; see "Cross-Subnet Configurations" (page 29).

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
```

CFS Package Status

If the cluster is using the Veritas Cluster File System (CFS), the system multi-node package `SG-CFS-pkg` must be running on all active nodes, and the multi-node packages for disk group and mount point must also be running on at least one of their configured nodes. If the cluster is using the Veritas Cluster Volume Manager for raw access to disk groups (that is, without CFS), `SG-CFS-pkg` must be running on all active nodes.

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CVM and CFS (http://www.hp.com/go/hpux-serviceguard-docs).

The following is an example of `cmviewcl` output showing status for these packages:

```
 cmviewcl -v -p SG-CFS-pkg

MULTI_NODE_PACKAGES

PACKAGE              STATUS       STATE        AUTO_RUN     SYSTEM
SG-CFS-pkg           up           running      enabled       yes

NODE_NAME       STATUS       SWITCHING
ftsys9           up           enabled

Script_Parameters:
ITEM      STATUS  MAX_RESTARTS  RESTARTS  NAME
Service    up      0             0          SG-CFS-vxconfigd
```

```
Service    up      5            0          SG-CFS-sgcvmd
Service    up      5            0          SG-CFS-vxfsckd
Service    up      0            0          SG-CFS-cmvxd
Service    up      0            0          SG-CFS-cmvxpingd

NODE_NAME      STATUS      SWITCHING
ftsys10         up          enabled

Script_Parameters:
ITEM      STATUS  MAX_RESTARTS  RESTARTS  NAME
Service    up      0            0          SG-CFS-vxconfigd
Service    up      5            0          SG-CFS-sgcvmd
Service    up      5            0          SG-CFS-vxfsckd
Service    up      0            0          SG-CFS-cmvxd
Service     up      0            0          SG-CFS-cmvxpingd
```

Status After Halting a Package

After we halt the failover package `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            service1
      Subnet        up                                  15.13.168.0
      Resource      up                                  /example/float

      Node_Switching_Parameters:
      NODE_TYPE     STATUS       SWITCHING    NAME
      Primary       up           enabled      ftsys9       (current)
      Alternate     up           enabled      ftsys10


  NODE          STATUS       STATE
  ftsys10       up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           28.1         lan0
    STANDBY      up           32.1         lan1

UNOWNED_PACKAGES

    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
    pkg2         down         halted      disabled     unowned
```

```
  Policy_Parameters:
  POLICY_NAME      CONFIGURED_VALUE
  Failover         configured_node
  Failback         manual


  Script_Parameters:
  ITEM         STATUS    NODE_NAME       NAME
  Resource     up        ftsys9          /example/float
  Subnet       up        ftsys9          15.13.168.0
  Resource     down        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 unowned, 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

If we use the following command

cmrunpkg -n ftsys9 pkg2

and then run cmviewcl -v, we'll see:

```
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                                    15.13.168.0
      Resource     up                                    /example/float

      Node_Switching_Parameters:
      NODE_TYPE    STATUS       SWITCHING    NAME
      Primary      up           enabled      ftsys9       (current)
      Alternate    up           enabled      ftsys10



    PACKAGE      STATUS       STATE        AUTO_RUN     NODE
```

```
   pkg2         up           running      disabled     ftsys9

     Policy_Parameters:
     POLICY_NAME      CONFIGURED_VALUE
     Failover         configured_node
     Failback         manual

     Script_Parameters:
     ITEM         STATUS       MAX_RESTARTS    RESTARTS     NAME
     Service      up           0               0            service2.1
     Subnet       up                                        15.13.168.0
     Node_Switching_Parameters:
     NODE_TYPE    STATUS       SWITCHING    NAME
     Primary      up           enabled      ftsys10
     Alternate    up           enabled      ftsys9     (current)

NODE         STATUS       STATE
ftsys10      up           running

     Network_Parameters:
     INTERFACE    STATUS       PATH         NAME
     PRIMARY        up           28.1         lan0
     STANDBY        up           32.1         lan1
```

Now `pkg2` is running on node `ftsys9`. Note that switching is still disabled.

Status After Auto Run is Enabled

The following command changes the `AUTO_RUN` package switching flag back to `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 `cmcld` daemon but no packages.

Status After Halting a Node

After we halt `ftsys10` with the following command

```
cmhaltnode  ftsys10
```

we'll see the following output from `cmviewcl`:

```
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 can be seen on both `ftsys9` and `ftsys10`.

Viewing Information about Unowned Packages

The following example shows packages that are currently unowned, that is, not running on any configured node. `cmviewcl` provides information on monitored resources 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 Information about System Multi-Node Packages

The following example shows a cluster that includes system multi-node packages as well as failover 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
SG-CFS-pkg   up           running
```

Checking Status of the Cluster File System (CFS)

If the cluster is using the cluster file system, you can check status with the `cfscluster` command, as in the example that follows. The `cfscluster status` command displays the status of the disk groups and mount points created only using legacy style of packaging, and not modular style of packaging.

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about CFS support: http://www.hp.com/go/hpux-serviceguard-docs.

`cfscluster status`

```
Node : ftsys9
Cluster Manager : up
CVM state : up (MASTER)

MOUNT POINT                TYPE   SHARED VOLUME   DISK GROUP       STATUS

/var/opt/sgtest/
tmp/mnt/dev/vx/dsk/
vg_for_cvm1_dd5/lvol1      regular lvol1   vg_for_cvm_dd5         MOUNTED

/var/opt/sgtest/
tmp/mnt/dev/vx/dsk/
vg_for_cvm1_dd5/lvol4      regular lvol4   vg_for_cvm_dd5         MOUNTED

Node : ftsys8
Cluster Manager : up
CVM state : up

MOUNT POINT               TYPE   SHARED VOLUME   DISK GROUP       STATUS

/var/opt/sgtest/
tmp/mnt/dev/vx/dsk/
vg_for_cvm1_dd5/lvol1     regular lvol1   vg_for_cvm_veggie_dd5  MOUNTED

/var/opt/sgtest/
tmp/mnt/dev/vx/dsk/
vg_for_cvm1_dd5/lvol4     regular lvol4   vg_for_cvm_dd5         MOUNTED
```

Status of the Packages in a Cluster File System

You can use `cmviewcl` to see the status of the package and the cluster file system on all nodes:

`cmviewcl -v -p SG-CFS-pkg`

```
MULTI_NODE_PACKAGES

PACKAGE STATUS STATE AUTO_RUN SYSTEM
SG-CFS-pkg up running enabled yes

NODE_NAME STATUS SWITCHING
soy up enabled

Script_Parameters:
ITEM STATUS MAX_RESTARTS RESTARTS NAME
Service up 0 0 SG-CFS-vxconfigd
Service up 5 0 SG-CFS-sgcvmd
Service up 5 0 SG-CFS-vxfsckd
Service up 0 0 SG-CFS-cmvxd
Service up 0 0 SG-CFS-cmvxpingd

NODE_NAME STATUS SWITCHING
tofu up enabled
```

```
Script_Parameters:
ITEM STATUS MAX_RESTARTS RESTARTS NAME
Service up 0 0 SG-CFS-vxconfigd
Service up 5 0 SG-CFS-sgcvmd
Service up 5 0 SG-CFS-vxfsckd
Service up 0 0 SG-CFS-cmvxd
Service up 0 0 SG-CFS-cmvxpingd
```

Status of CFS Modular Disk Group and Mount Point Packages

To see the status of CFS modular disk group and mount point packages, use the `cmviewcl` command. For example if the modular package is `mpdg1`, enter:

```
cmviewcl -v -p mpdg1
```

```
MULTI_NODE_PACKAGES

  PACKAGE        STATUS           STATE            AUTO_RUN     SYSTEM
  mpdg1          up               running          enabled      no

    NODE_NAME      STATUS           STATE            SWITCHING
    node1          up               running          enabled

      Dependency_Parameters:
      DEPENDENCY_NAME         SATISFIED
      SG-CFS-pkg              yes

    NODE_NAME      STATUS           STATE            SWITCHING
    node2          up               running          enabled

      Dependency_Parameters:
      DEPENDENCY_NAME         SATISFIED
      SG-CFS-pkg              yes

    Other_Attributes:
    ATTRIBUTE_NAME                          ATTRIBUTE_VALUE
    Style                                   modular
    Priority                                no_priority
```

Status of Legacy CVM Disk Group Packages

To see the status of the legacy CVM disk group, use the `cfsdgadm display` command. For example, for the disk group `logdata`, enter:

```
cfsdgadm display -v logdata
```

```
NODE NAME    ACTIVATION MODE
ftsys9                  sw (sw)
MOUNT POINT     SHARED VOLUME TYPE
ftsys10                 sw (sw)
MOUNT POINT SHARED VOLUME TYPE
...
```

To see which package is monitoring a disk group, use the `cfsdgadm show_package` command. For example, for the disk group `logdata`, enter:

```
cfsdgadm show_package logdata
```

```
SG-CFS-DG-1
```

Status of Legacy CFS Mount Point Packages

To see the status of the legacy CFS mount point package, use the `cfsmntadm display` command. For example, for the mount point `/tmp/logdata/log_files`, enter:

```
cfsmntadm display -v /tmp/logdata/log_files
```

```
Mount Point : /tmp/logdata/log_files
Shared Volume : lvol1
Disk Group : logdata
```

To see which package is monitoring a mount point, use the `cfsmntadm show_package` command. For example, for the disk group `logdata`:

```
cfsmntadm show_package /tmp/logdata/log_files
SG-CFS_MP-1
```

Checking the Cluster Configuration and Components

Serviceguard provides tools that allow you to check the soundness of the cluster configuration, and the health of its components. In past releases, much of this was done by `cmcheckconf (1m)` and/or `cmapplyconf (1m)` and could be done only when you were changing the configuration of the cluster or packages; see "Verifying the Cluster Configuration " (page 188) and "Verifying and Applying the Package Configuration" (page 245).

As of Serviceguard A.11.20, these commands perform additional checks, and a new command, `cmcompare (1m)` allows you to compare the contents and characteristics of cluster-wide files to make sure they are consistent. In addition, you can check configuration of the cluster and all of its packages at any time by running `cmcheckconf (1m)` without arguments (or with `-v`; see below). These checks help you to ensure that packages will start up and fail over successfully.

Specifically, the following capabilities are new as of Serviceguard A.11.20.

NOTE:

- All of the checks below are performed when you run `cmcheckconf` without any arguments (or with only `-v`, with or without `-k` or `-K`). `cmcheckconf` validates the current cluster and package configuration, including external scripts and pre-scripts for modular packages, and runs `cmcompare` to check file consistency across nodes. (This new version of the command also performs all of the checks that were done in previous releases.) See "Checking Cluster Components" (page 261) for details.

  You may want to set up a `cron (1m)` job to run `cmcheckconf` regularly. See "Setting up Periodic Cluster Verification" (page 263).

- These new checks *are not* done for legacy packages. For information about legacy and modular packages, see Chapter 6 (page 216).

- LVM volume groups:
  - Check that each volume group contains the same physical volumes on each node
  - Check that each node has a working physical connection to the physical volumes
  - Check that volume groups used in modular packages are cluster-aware
- LVM logical volumes
  - Check that file systems have been built on the logical volumes identified by the *`fs_name`* parameter (page 238) in the cluster's packages.
- VxVM disk groups:
  - Check that each disk group contains the same disks on each node
  - Check that each node has a working physical connection to the disks
- File consistency:
  - Check that files including the following are consistent across all nodes:
    - `/etc/hosts` (must contain all IP addresses configured into the cluster)
    - `/etc/nsswitch.conf`
    - `/etc/services`
    - package control scripts for legacy packages (if you specify them)
    - `/etc/cmcluster/cmclfiles2check`
    - `/etc/cmcluster/cmignoretypes.conf`
    - `/etc/cmcluster/cmknowncmds`
    - `/etc/cmcluster/cmnotdisk.conf`
    - user-created files (if you specify them)

Checking Cluster Components

The following table shows each component that you can check, the command or tool to use, and where to find further information.

NOTE: The table includes all the checks available as of A.11.20, not just the new ones.

Table 12 Verifying Cluster Components

Component (Context)	Tool or Command; More Information	Comments
Volume groups (cluster)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying the Cluster Configuration " (page 188).	Checked for: • existence • availability on each node • same physical volumes on each node • physical volumes connected on each node
Volume groups (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Checked only on nodes configured to run the package.
LVM logical volumes (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Checked for modular packages only, as part of package validation (`cmcheckconf -P`)
LVM physical volumes (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)`	Checked for modular packages only, as part of package validation (`cmcheckconf -P`)
LANs (cluster)	`cmviewcl (1m)`, `cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Reviewing Cluster and Package Status" (page 248).	Standby status is checked for each single NIC; `cmviewcl` also reports APA status. `cmviewcl` flags offline standby in an existing cluster, but not for a cluster that is being created.
Quorum Server (cluster)	`cmcheckconf (1m)`, `cmapplyconf (1m)`.	Commands check that the quorum server, if used, is running and all nodes are authorized to access it; and, if more than one IP address is specified, that the quorum server is reachable from all nodes through both the IP addresses.
Lock disk(s) (cluster)	`cmcheckconf (1m)`, `cmapplyconf (1m)`	Commands check that the disk(s) are accessible from all nodes, and that the lock volume group(s) are activated on at least one node.
Lock LUN (cluster)	`cmcheckconf (1m)`, `cmapplyconf (1m)`	Commands check that all nodes are be accessing the same physical device and that the lock LUN device file is a block device file

Table 12 Verifying Cluster Components *(continued)*

Component (Context)	Tool or Command; More Information	Comments
File consistency (cluster)	`cmcheckconf (1m)`, `cmcompare (1m)`. **IMPORTANT:** See the manpage for differences in return codes from `cmcheckconf` without options versus `cmcheckconf -C`	To check file consistency across all nodes in the cluster, do the following: 1. Customize `/etc/cmcluster/cmfiles2check` 2. Distribute it to all nodes using `cmsysnc (1m)` 3. run `cmcheckconf -C` For a subset of nodes, or to check only specific characteristics such as ownership, content, etc., use `cmcompare (1m)`.
VxVM disk groups (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Commands check that each node has a working physical connection to the disks.
Mount points (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Commands check that the mount-point directories specified in the package configuration file exist on all nodes that can run the package.
Service commands (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Commands check that files specified by service commands exist and are executable. Service commands whose paths are nested within an unmounted shared file system are not checked.
IP addresses (cluster)	`cmcheckconf (1m)`, `cmapplyconf (1m)`	Commands check that all IP addresses configured into the cluster are in each node's `/etc/hosts`.
Package IP addresses (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	
File systems (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	For LVM only, commands check that file systems have been built on the logical volumes identified by the *`fs_name`* parameter (page 238).
EMS resources (package)	`cmcheckconf (1m)`, `cmapplyconf (1m)` See also "Verifying and Applying the Package Configuration" (page 245).	Commands check that configured resources are available on each node that can run the package.
External scripts and pre-scripts (modular package)	`cmcheckconf (1m)`, `cmapplyconf (1m)`	A non-zero return value from any script will cause the commands to fail.

Setting up Periodic Cluster Verification

You can use `cron (1m)` to run cluster verification at a fixed interval. Specify the commands to run in a `crontab` file (see `crontab (1)`).

NOTE: The job must run on one of the nodes in the cluster. Because only the root user can run cluster verification, and `cron (1m)` sets the job's user and group ID's to those of the user who submitted the job, you must edit the file `/var/spool/cron/crontabs/root` as the root user.

Example

The short script that follows runs cluster verification and sends an email to `admin@hp.com` when verification fails.

```
#!/bin/sh

cmcheckconf -v >/tmp/cmcheckconf.output
if (( $? != 0 ))
     then
mailx -s "Cluster verification failed" admin@hp.com 2>&1 </tmp/cmcheckconf.output

fi
```

To run this script from `cron`, you would create the following entry in `/var/spool/cron/crontabs/root`:

```
0 8,20 * * * verification.sh
```

See the `cron (1m)` manpage for more information.

Limitations

Serviceguard *does not* check the following conditions:

- Access Control Policies properly configured (see "Controlling Access to the Cluster" (page 183) for information about Access Control Policies)
- File systems configured to mount automatically on boot (that is, Serviceguard does not check `/etc/fstab`)
- Shared volume groups configured to activate on boot
- Volume group major and minor numbers unique
- Redundant storage paths functioning properly
- Kernel parameters and driver configurations consistent across nodes
- Mount point overlaps (such that one file system is obscured when another is mounted)
- Unreachable DNS server
- Consistency of settings in `.rhosts` and `/var/admin/inetd.sec`
- Consistency across cluster of major and minor numbers device-file numbers
- Nested mount points
- Staleness of mirror copies

Managing the Cluster and Nodes

This section covers 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
- Halting a Node or the Cluster while Keeping Packages Running (page 267)

In Serviceguard A.11.16 and later, these tasks can be performed by non-root users with the appropriate privileges, except where specifically noted. See "Controlling Access to the Cluster" (page 183) 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 as shown below, to start the cluster.

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 following command starts all nodes configured in the cluster and verifies the network information:

```
cmruncl
```

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.

Use the `-v` (verbose) option to display the greatest number of messages.

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
```

△ **CAUTION:** 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 as shown below, to bring a configured node up within a running cluster.

Using Serviceguard Commands to Add Previously Configured Nodes to a Running Cluster

Use the `cmrunnode` command to join 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. The -v (verbose) option prints out all the messages:

```
cmrunnode -v ftsys8
```

By default, cmrunnode 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. Packages may be started, depending on the package configuration; see "*node_name*" (page 223)). 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 Participation in a Running Cluster

You can use Serviceguard Manager, or Serviceguard commands as shown below, to remove nodes from active participation in a cluster. This operation halts 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.

Halting a node is a convenient way of bringing it 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 Failover Package " (page 273).

To return a node to the cluster, use cmrunnode.

NOTE: HP recommends that you remove a node from participation in the cluster (by running cmhaltnode as shown below, or Halt Node in Serviceguard Manager) before running the HP-UX shutdown command, especially in cases in which a packaged application might have trouble during shutdown and not halt cleanly.

Use cmhaltnode 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 an adoptive node The -v (verbose) option prints out messages:

```
cmhaltnode -f -v ftsys9
```

This halts any packages running on the node ftsys9 by executing the halt instructions in each package's master control script. ftsys9 is halted and the packages start on their adoptive node.

Halting the Entire Cluster

You can use Serviceguard Manager, or Serviceguard commands as shown below, to halt a running cluster.

Use cmhaltcl 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. You can use the command on any running node, for example:

```
cmhaltcl -f -v
```

This halts all the cluster nodes.

Automatically Restarting the Cluster

You can configure your cluster to automatically restart after an event, such as a long-term power failure, which brought down all nodes in the cluster. This is done by setting *AUTOSTART_CMCLD* to 1 in the /etc/rc.config.d/cmcluster file.

Halting a Node or the Cluster while Keeping Packages Running

There may be circumstances in which you want to do maintenance that involves halting a node, or the entire cluster, without halting or failing over the affected packages. Such maintenance might consist of anything short of rebooting the node or nodes, but a likely case is networking changes that will disrupt the heartbeat.

New command options in Serviceguard A.11.20 (collectively known as Live Application Detach (LAD)) allow you to do this kind of maintenance while keeping the packages running. The packages are no longer monitored by Serviceguard, but the applications continue to run. Packages in this state are called detached packages.

When you have done the necessary maintenance, you can restart the node or cluster, and normal monitoring will resume on the packages.

NOTE: Keep in mind that the purpose of the LAD capabilities is to allow you do maintenance on one or more nodes, or the entire cluster. If you want to do maintenance on individual packages, or on elements of the cluster configuration that affect only one package, or a few packages, you should probably use package maintenance mode; see "Maintaining a Package: Maintenance Mode" (page 273).

What You Can Do

You can do the following.

- Halt a node (`cmhaltnode (1m)` with the `-d` option) without causing its running packages to halt or fail over.

  Until you restart the node (`cmrunnode (1m)`) these packages are detached — not being monitored by Serviceguard.

- Halt the cluster (`cmhaltcl (1m)` with the `-d` option) without causing its running packages to halt.

  Until you restart the cluster (`cmruncl (1m)`) these packages are detached — not being monitored by Serviceguard.

- Halt a detached package, including instances of detached multi-node packages.
- Restart normal package monitoring by restarting the node (`cmrunnode`) or the cluster (`cmruncl`).

Rules and Restrictions

The following rules and restrictions apply.

- All the nodes in the cluster must be running A.11.20.
- All the configured cluster nodes must be reachable by an available network (not necessarily a network that is configured into the cluster).
- You must be root user (superuser) to halt or start a node or cluster with Live Application Detach, and to halt a detached package.
- Live Application Detach is not supported for SGeRAC clusters configured with SLVM or ASM.
- Live Application Detach is not supported for clusters using the Veritas Cluster Volume Manager (CVM) or Cluster File System (CFS).
- Live Application Detach is supported only for modular failover packages and modular multi-node packages.
  - You cannot detach legacy packages, but you can halt them and then detach the modular packages.
  - You cannot use Live Application Detach if system multi-node packages (such as `SG-CFS-pkg`) are configured in the cluster.

See Chapter 6 (page 216) for more information about package types.

- You cannot detach a package that is in maintenance mode, and you cannot place a package into maintenance mode if any of its dependent packages are detached.

  For more information about maintenance mode, see"Maintaining a Package: Maintenance Mode" (page 273). For more information about dependencies, see "About Package Dependencies" (page 128).

- You cannot make configuration changes to a cluster in which any packages are detached.

  `cmapplyconf (1m)` will fail.

- You cannot halt detached packages while the cluster is down.

  If you have halted a node and detached its packages, you can log in as superuser on any other node still running in the cluster and halt any of the detached packages. But if you have halted the cluster, you must restart it, re-attaching the packages, before you can halt any of the packages

- If you are using LVM, all the cluster nodes must have Base LVM version B.11.31.0909 or later.
- HPVM shared volume groups are not supported with Live Application detach.
- `cmeval (1m)` does not support Live Application Detach.

  See "Previewing the Effect of Cluster Changes" (page 279) for more information about `cmeval`.

- In preview mode (`-t`) `cmrunnode` and `cmruncl` can provide only a partial assessment of the effect of re-attaching packages.

  The assessment may not accurately predict the placement of packages that depend on the packages that will be re-attached. For more information about preview mode, see "Previewing the Effect of Cluster Changes" (page 279).

- `cmmodpkg -e -t` is not supported for a detached package.
- You cannot run a package that has been detached.

  This could come up if you detect that a package has failed while detached (and hence not being monitored by Serviceguard). Before you could restart the package on another node, you would need to run `cmhaltpkg (1m)` to halt the package on the node where it is detached.

- You cannot halt a package that is in a transitory state such as `STARTING` or `HALTING`.

  For more information about package states, see "Package Status and State" (page 249).

- You cannot detach a package that has relocatable IP addresses on a subnet that is not configured into the cluster.

  Such addresses will not be handled properly when the package is attached or re-attached.

IMPORTANT: HP does not recommend such configurations. See "*ip_subnet*" (page 231).

- You cannot detach packages when the HP-UX Clustered I/O Congestion Control feature is enabled.
- As of the date of this manual, you cannot detach ECMT-based packages.

Additional Points To Note

Keep the following points in mind.

- When packages are detached, they continue to run, but without high availability protection.

  Serviceguard does not detect failures of components of detached packages, and packages are not failed over. Serviceguard will not provide local LAN failover of IP addresses if the node has been halted in detach mode.

IMPORTANT: This means that you will need to detect any errors that occur while the package is detached, and take corrective action by running `cmhaltpkg` to halt the detached package and `cmrunpkg (1m)` to restart the package on another node.

- `cmviewcl (1m)` reports the status and state of detached packages as `detached`.

  This is true even if a problem has occurred since the package was detached and some or all of the package components are not healthy or not running.

- Because Serviceguard assumes that a detached package has remained healthy, the package is considered to be `UP` for dependency purposes.

  This means, for example, that if you halt `node1`, detaching `pkgA`, and `pkgB` depends on `pkgA` to be `UP` on `ANY_NODE`, `pkgB` on `node2` will continue to run (or can start) while `pkgA` is detached. See "About Package Dependencies" (page 128) for more information about dependencies.

- As always, packages cannot start on a halted node or in a halted cluster.
- When you restart a node or cluster whose packages have been detached, the packages are re-attached; that is, Serviceguard begins monitoring them again.

  At this point, Serviceguard checks the health of the packages that were detached and takes any necessary corrective action — for example, if a failover package has in fact failed while it was detached, Serviceguard will halt it and restart it on another eligible node.

CAUTION: Serviceguard *does not* check LVM volume groups and mount points when re-attaching packages.

- The `detached` state and status could appear to persist across a reboot.

  If a node reboots while packages are detached (or detaching, or re-attaching), and package components are left in an inconsistent state, the appropriate package halt scripts will run to clean things up when the node comes back up. But `cmviewcl` will continue to show the packages as `detached`. Either `cmhaltpkg` or `cmrunnonde (1m)` will reset the packages' state and status.

- If you halt a package and disable it before running `cmhaltcl -d` to detach other packages running in the cluster, *`auto_run`* will be automatically re-enabled for this package when the cluster is started again, forcing the package to start.

  To prevent this behavior and keep the package halted and disabled after the cluster restarts, change *`auto_run`* to `no` in the package configuration file (page 224), and re-apply the package, before running `cmhaltcl -d`.

- If an IP address is switched to the standby LAN because of a failure of on the primary LAN before a node is halted in detached mode, and if the failure is detected as an IP-only failure (meaning that the primary LAN was failed at the IP level only) then the IP address will remain on the standby LAN even after the node is restarted via `cmrunnode`. This will also happen

if the IP address is switched to the standby LAN and *NETWORK_AUTO_FAILBACK* cluster parameter is set to FALSE.

If the primary LAN recovers while the node is halted and you want the IP address to fail back to the primary LAN, run cmmodnet –e to re-enable the primary LAN interface and trigger the failback.

Halting a Node and Detaching its Packages

To halt a node and detach its packages, proceed as follows.

1. Make sure that the conditions spelled out under "Rules and Restrictions" (page 267) are met.
2. Halt any packages that do not qualify for Live Application Detach, such as legacy and system multi-node packages.

   For example:

   ```
   cmhaltpkg -n node1 legpak1 legpak2
   ```

   NOTE: If you do not do this, the cmhaltnode in the next step will fail.

3. Halt the node with the -d (detach) option:

   ```
   cmhaltnode -d node1
   ```

   NOTE: -d and -f are mutually exclusive. See cmhaltnode (1m) for more information.

To re-attach the packages, restart the node:

```
cmrunnode -n node1
```

Halting a Detached Package

To halt a package that is detached on node1, proceed as follows.

1. Log in as superuser on another node that is still running in the cluster.
2. Halt the package; for example:

   ```
   cmhaltpkg node1 pkg1
   ```

Halting the Cluster and Detaching its Packages

1. Make sure that the conditions spelled out under "Rules and Restrictions" (page 267) are met.
2. Halt any packages that do not qualify for Live Application Detach, such as legacy and system multi-node packages.

   For example:

   ```
   cmhaltpkg legpak1 legpak2 legpak3 smnp1
   ```

   NOTE: If you do not do this, the cmhaltcl in the next step will fail.

3. Halt the cluster with the -d (detach) option:

   ```
   cmhaltcl -d
   ```

   NOTE: -d and -f are mutually exclusive. See cmhaltcl (1m) for more information.

To re-attach the packages, restart cluster:

```
cmrunnode node1
```

Example: Halting the Cluster for Maintenance on the Heartbeat Subnets

Suppose that you need to do networking maintenance that will disrupt all the cluster's heartbeat subnets, but it is essential that the packages continue to run while you do it. In this example we'll

assume that packages pkg1 through pkg5 are unsupported for Live Application Detach, and pkg6 through pkg*n* are supported.

Proceed as follows:

1. Halt all the unsupported packages:

   cmhaltpkg pkg1 pkg2 pkg3 pkg4 pkg5

2. Halt the cluster, detaching the remaining packages:

   cmhaltcl -d

3. Upgrade the heartbeat networks as needed.
4. Restart the cluster, automatically re-attaching pkg6 through pkg*n* and starting any other packages that have *auto_run* (page 224) set to yes in their package configuration file:

   cmruncl

5. Start the remaining packages; for example

   cmmodpkg -e pkg1 pkg2 pkg3 pkg4 pkg5

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
- Maintaining a package using maintenance mode

Non-root users with the appropriate privileges can perform these tasks. See "Controlling Access to the Cluster" (page 183) for information about configuring access.

You can use Serviceguard Manager or the Serviceguard command line to perform these tasks.

Starting a Package

Ordinarily, when a cluster starts up, the packages configured as part of the cluster will start up on their configured nodes. 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.

If any package has a configured dependency on another package, Serviceguard will start them in order, ensuring that a package will not start until its dependency is met.

You can use Serviceguard Manager, or Serviceguard commands as shown below, to start a package.

The cluster must be running, and if the package is dependent on other packages, those packages must be either already running, or started by the same command that starts this package (see the section that follows, and "About Package Dependencies" (page 128).)

Starting a Package that Has Dependencies

Before starting a package, it is a good idea to use the cmviewcl command to check for package dependencies.

You cannot start a package unless all the packages that it depends on are running. If you try, you'll see a Serviceguard message telling you why the operation failed, and the package will not start.

If this happens, you can repeat the run command, this time including the package(s) this package depends on; Serviceguard will start all the packages in the correct order.

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. For example, to start a failover package:

```
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.

Starting the Special-Purpose CVM and CFS Packages

Use CFS administration commands to start the special-purpose multi-node packages used with CFS. For example, to start the special-purpose multi-node package for the disk group package (`SG-CFS-DG-id#`), use the `cfsdgadm` command. To start the special-purpose multi-node package for the mount package (`SG-CFS-MP-id#`) use the `cfsmntadm` command. Check to see if your package has a dependency; before you can start your dependent package, you must start all the packages it depends on.

Halting a Package

You halt a Serviceguard package when you want to bring the package out of use but want the node to continue running in the cluster. You can halt a package using Serviceguard Manager or on the Serviceguard command line.

Halting a package has a different effect from halting the node. When you halt the node, its failover packages may switch to adoptive nodes (assuming that switching is enabled for them); when you halt a failover package, it is disabled from switching to another node, and must be restarted manually on another node or on the same node.

System multi-node packages run on all cluster nodes simultaneously; halting these packages stops them running on all nodes. A multi-node package can run on several nodes simultaneously; you can halt it on all the nodes it is running on, or you can specify individual nodes.

Halting a Package that Has Dependencies

Before halting a package, it is a good idea to use the `cmviewcl` command to check for package dependencies.

You cannot halt a package unless all the packages that depend on it are down. If you try, you'll see a Serviceguard message telling you why the operation failed, and the package will remain up.

If this happens, you can repeat the halt command, this time including the dependent package(s); Serviceguard will halt all the packages in the correct order. First, use `cmviewcl` to be sure that no *other* running package has a dependency on any of the packages you are halting.

You can use Serviceguard Manager, or Serviceguard commands as shown below, to halt a package.

Using Serviceguard Commands to Halt a Package

Use the `cmhaltpkg` command to halt a package, as follows:

```
cmhaltpkg pkg1
```

This halts `pkg1`, and, if `pkg1` is a failover package, also disables it from switching to another node.

You cannot halt a package unless all packages that depend on it are down. If you try, Serviceguard will take no action, except to send a message indicating that not all dependent packages are down. Before you halt a system multi-node package, or halt all instances of a multi-node package, halt any packages that depend on them

Moving a Failover Package

You can use Serviceguard Manager, or Serviceguard commands as shown below, to move a failover package from one node to another.

Using Serviceguard Commands to Move a Running Failover Package

Before you move a failover package to a new node, it is a good idea to run `cmviewcl -v -l package` and look at dependencies. If the package has dependencies, be sure they can be met on the new node.

To move the package, first halt it where it is running using the `cmhaltpkg` command. This action not only halts the package, but also disables package switching.

After it halts, run the package on the new node using the `cmrunpkg` command, then re-enable switching as described under "Using Serviceguard Commands to Start a Package" (page 272).

Changing Package Switching Behavior

There are two options to consider:

- Whether the package can switch (fail over) or not.
- Whether the package can switch to a particular node or not.

For failover packages, if package switching is set to `NO` the package cannot move to any other node; if node switching is set to `NO`, the package cannot move to that particular node. For multi-node packages, if package switching is set to `NO`, the package cannot start on a new node joining the cluster; if node switching is set to `NO`, the package cannot start on that node.

Both node switching and package switching can be changed dynamically while the cluster is running. The initial setting for package switching is determined by the *`auto_run`* parameter, which is set in the package configuration file (page 224). If *`auto_run`* is set to `yes`, then package switching is enabled when the package first starts. The initial setting for node switching is to allow switching to all nodes that are configured to run the package.

You can use Serviceguard Manager to change package switching behavior, or Serviceguard commands as shown below.

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 prevent it from starting up on another node, enter the following:

```
cmmodpkg -d pkg1
```

This does not halt the package, but will prevent it from starting up elsewhere.

You can disable package switching to particular nodes by using the `-n` option of the `cmmodpkg` command. The following prevents `pkg1` from switching 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, change the *`auto_run`* flag in the package configuration file, then re-apply the configuration. (See "Reconfiguring a Package on a Running Cluster " (page 297).)

Maintaining a Package: Maintenance Mode

Serviceguard A.11.20 provides two ways to perform maintenance on components of a modular, failover package while the package is running. (See Chapter 6 (page 216) for information about package types and modules.) These two methods are called maintenance mode and partial-startup maintenance mode.

NOTE: If you need to do maintenance that requires halting a node, or the entire cluster, you should consider Live Application Detach; see "Halting a Node or the Cluster while Keeping Packages Running" (page 267).

- Maintenance mode is chiefly useful for modifying networks and EMS resources used by a package while the package is running.

  See "Performing Maintenance Using Maintenance Mode" (page 276).

- Partial-startup maintenance mode allows you to work on package services, file systems, and volume groups.

  See "Performing Maintenance Using Partial-Startup Maintenance Mode" (page 276).

- Neither maintenance mode nor partial-startup maintenance mode can be used for legacy packages, multi-node packages, or system multi-node packages.
- Package maintenance does not alter the configuration of the package, as specified in the package configuration file.

  For information about reconfiguring a package, see "Reconfiguring a Package" (page 297).

NOTE: In order to run a package in partial-startup maintenance mode, you must first put it in maintenance mode. This means that packages in partial-startup maintenance mode share the characteristics described below for packages in maintenance mode, and the same rules and dependency rules apply. Additional rules apply to partial-startup maintenance mode, and the procedure involves more steps, as explained underPerforming Maintenance Using Partial-Startup Maintenance Mode.

Characteristics of a Package Running in Maintenance Mode or Partial-Startup Maintenance Mode

Serviceguard treats a package in maintenance mode differently from other packages in important ways. The following points apply to a package running in maintenance mode:

- Serviceguard ignores failures reported by package services, subnets, EMS resources, and file systems; these will not cause the package to fail.

  NOTE: But a failure in the package control script *will* cause the package to fail. The package will also fail if an external script (or pre-script) cannot be executed or does not exist.

- The package will not be automatically failed over, halted, or started.
- A package in maintenance mode still has its configured (or default) weight, meaning that its weight, if any, is counted against the node's capacity; this applies whether the package is up or down. (See "About Package Weights" (page 135) for a discussion of weights and capacities.)
- Node-wide and cluster-wide events affect the package as follows:
  - If the node the package is running on is halted or crashes, the package will no longer be in maintenance mode but will not be automatically started.
  - If the cluster is halted or crashes, the package will not be in maintenance mode when the cluster comes back up. Serviceguard will attempt to start it if `auto_run` is set to `yes` in the package configuration file.
- If `node_fail_fast_enabled` (page 224) is set to `yes`, Serviceguard *will not* halt the node under any of the following conditions:
  - Subnet failure
  - EMS resource failure
  - A script does not exist or cannot run because of file permissions

- A script times out
- The limit of a restart count is exceeded

Rules for a Package in Maintenance Mode or Partial-Startup Maintenance Mode

IMPORTANT: See the latest Serviceguard release notes for important information about version requirements for package maintenance.

- The package must have package switching disabled before you can put it in maintenance mode.
- You can put a package in maintenance mode only on one node.
  - The node must be active in the cluster and must be eligible to run the package (on the package's *node_name* list).
  - If the package is not running, you must specify the node name when you run `cmmodpkg (1m)` to put the package in maintenance mode.
  - If the package is running, you can put it into maintenance only on the node on which it is running.
  - While the package is in maintenance mode on a node, you can run the package only on that node.
- You cannot put a package in maintenance mode, or take it out maintenance mode, if doing so will cause another running package to halt.
- Since package failures are ignored while in maintenance mode, you can take a running package out of maintenance mode only if the package is healthy.

  Serviceguard checks the state of the package's services, subnets and EMS resources to determine if the package is healthy. If it is not, you must halt the package before taking it out of maintenance mode.
- You cannot do online configuration as described under "Reconfiguring a Package" (page 297).
- You cannot configure new dependencies involving this package; that is, you cannot make it dependent on another package, or make another package depend on it. See also "Dependency Rules for a Package in Maintenance Mode or Partial-Startup Maintenance Mode " (page 276).
- You cannot use the `-t` option of any command that operates on a package that is in maintenance mode; see "Previewing the Effect of Cluster Changes" (page 279) for information about the `-t` option.

Additional Rules for Partial-Startup Maintenance Mode

- You must halt the package before taking it out of partial-startup maintenance mode.
- To run a package normally after running it in partial-startup maintenance mode, you must take it out of maintenance mode, and then restart it.

Dependency Rules for a Package in Maintenance Mode or Partial-Startup Maintenance Mode

You cannot configure new dependencies involving a package running in maintenance mode, and in addition the following rules apply (we'll call the package in maintenance mode `pkgA`).

- The packages that depend on `pkgA` must be down with package switching disabled when you place `pkgA` in maintenance mode. This applies to all types of dependency (including exclusionary dependencies) as described under "About Package Dependencies" (page 128).
  - You cannot enable a package that depends on `pkgA`.
  - You cannot run a package that depends on `pkgA`, unless the dependent package itself is in maintenance mode.
- Dependency rules governing packages that `pkgA` depends on to be `UP` are bypassed so that these packages can halt and fail over as necessary while `pkgA` is in maintenance mode.
- If both packages in a dependency relationship are in maintenance mode, dependency rules are ignored for those two packages.

  For example, both packages in an exclusionary dependency can be run and halted in maintenance mode at the same time.

Performing Maintenance Using Maintenance Mode

You can put a package in maintenance mode, perform maintenance, and take it out of maintenance mode, whether the package is down or running.

This mode is mainly useful for making modifications to networking or EMS resources. To modify other components of the package, such as services or storage, follow the additional rules and instructions under "Performing Maintenance Using Partial-Startup Maintenance Mode" (page 276).

If you want to reconfigure the package (using `cmapplyconf (1m)`) see "Reconfiguring a Package" (page 297) and "Allowable Package States During Reconfiguration " (page 300).

Procedure

Follow these steps to perform maintenance on a package's networking or EMS components.

In this example, we'll call the package `pkg1` and assume it is running on `node1`.

1. Place the package in maintenance mode:

   ```
   cmmodpkg -m on -n node1 pkg1
   ```

2. Perform maintenance on the networks or resources and test manually that they are working correctly.

   NOTE: If you now run `cmviewcl`, you'll see that the `STATUS` of `pkg1` is `up` and its `STATE` is `maintenance`.

3. If everything is working as expected, take the package out of maintenance mode:

   ```
   cmmodpkg -m off pkg1
   ```

Performing Maintenance Using Partial-Startup Maintenance Mode

To put a package in partial-startup maintenance mode, you put it in maintenance mode, then restart it, running only those modules that you *will not* be working on.

Procedure

Follow this procedure to perform maintenance on a package. In this example, we'll assume a package `pkg1` is running on `node1`, and that we want to do maintenance on the package's services.

1. Halt the package:

   cmhaltpkg pkg1

2. Place the package in maintenance mode:

   cmmodpkg -m on -n node1 pkg1

   NOTE: The order of the first two steps can be reversed.

3. Run the package in maintenance mode.

   In this example, we'll start `pkg1` such that only the modules up to and including the `package_ip` module are started. (See "Package Modules and Parameters" (page 219) for a list of package modules. The modules used by a package are started in the order shown near the top of its package configuration file.)

   cmrunpkg -m sg/package_ip pkg1

4. Perform maintenance on the services and test manually that they are working correctly.

   NOTE: If you now run `cmviewcl`, you'll see that the `STATUS` of `pkg1` is `up` and its `STATE` is `maintenance`.

5. Halt the package:

   cmhaltpkg pkg1

   NOTE: You can also use `cmhaltpkg -s`, which stops the modules started by `cmrunpkg -m` — in this case, all the modules up to and including `package_ip`.

6. Run the package to ensure everything is working correctly:

   cmrunpkg pkg1

   NOTE: The package is still in maintenance mode.

7. If everything is working as expected, bring the package out of maintenance mode:

   cmmodpkg -m off pkg1

8. Restart the package:

   cmrunpkg pkg1

Excluding Modules in Partial-Startup Maintenance Mode

In the example above, we used `cmrunpkg -m` to run all the modules up to and including `package_ip`, but none of those after it. But you might want to run the entire package apart from the module whose components you are going to work on. In this case you can use the `-e` option:

cmrunpkg -e sg/service pkg1

This runs all the package's modules *except* the services module.

You can also use `-e` in combination with `-m`. This has the effect of starting all modules up to and including the module identified by `-m`, except the module identified by `-e`. In this case the excluded (`-e`) module must be earlier in the execution sequence (as listed near the top of the package's configuration file) than the `-m` module. For example:

cmrunpkg -m sg/services -e sg/package_ip pkg1

NOTE: The full execution sequence for starting a package is:

1. The master control script itself
2. External pre-scripts
3. Volume groups
4. File systems
5. Package IPs
6. External scripts
7. Services

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 13 shows the required cluster state for many kinds of changes.

Table 13 Types of Changes to the Cluster Configuration

Change to the Cluster Configuration	Required Cluster State
Add a new node	All systems configured as members of this cluster must be running.
Delete a node	A node can be deleted even though it is unavailable or unreachable.
Add a volume group	Cluster can be running.
Delete a volume group	Cluster can be running. Packages that use the volume group will not be able to start again until their configuration is modified.
Change Maximum Configured Packages	Cluster can be running.
Change Quorum Server Configuration	Cluster can be running; see the NOTE below this table and "What Happens when You Change the Quorum Configuration Online" (page 47).
Change Cluster Lock Configuration (LVM lock disk)	Cluster can be running; see "Updating the Cluster Lock Configuration" (page 281), "What Happens when You Change the Quorum Configuration Online" (page 47), and the NOTE below this table.
Change Cluster Lock Configuration (lock LUN)	Cluster can be running. See "Updating the Cluster Lock LUN Configuration Online" (page 282), "What Happens when You Change the Quorum Configuration Online" (page 47), and the NOTE below this table.
Add NICs and their IP addresses, if any, to the cluster configuration	Cluster can be running. See "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).
Delete NICs and their IP addresses, if any, from the cluster configuration	Cluster can be running. "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284). If removing the NIC from the system, see "Removing a LAN or VLAN Interface from a Node" (page 287).
Change the designation of an existing interface from `HEARTBEAT_IP` to `STATIONARY_IP`, or *vice versa*	Cluster can be running. See "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284).
Change an interface from IPv4 to IPv6, or *vice versa*	Cluster can be running. See "Changing the Cluster Networking Configuration while the Cluster Is Running" (page 284)
Reconfigure IP addresses for a NIC used by the cluster	Must delete the interface from the cluster configuration, reconfigure it, then add it back into the cluster configuration.

Table 13 Types of Changes to the Cluster Configuration *(continued)*

Change to the Cluster Configuration	Required Cluster State
	See "What You Must Keep in Mind" (page 284). Cluster can be running throughout.
Change *NETWORK_FAILURE_DETECTION* parameter (see "Monitoring LAN Interfaces and Detecting Failure: Link Level" (page 66))	Cluster can be running.
Change *NETWORK_AUTO_FAILBACK* parameter	Cluster can be running.
Change *NETWORK_POLLING_INTERVAL*	Cluster can be running.
Change IP Monitor parameters: *SUBNET*, *IP_MONITOR*, *POLLING TARGET*	Cluster can be running. See the entries for these parameters under "Cluster Configuration Parameters " (page 105) for more information.
Change *MEMBER_TIMEOUT* and *AUTO_START_TIMEOUT*	Cluster can be running, except in CVM environment; see the NOTE below this table.
Change Access Control Policy	Cluster and package can be running.
Change capacity and weight parameters.	Cluster can be running. A change that would cause a running package to fail will trigger a warning, giving you a chance to cancel (unless you use `cmapplyconf -f`).

NOTE: If you are using CVM or CFS, you cannot change *MEMBER_TIMEOUT* or *AUTO_START_TIMEOUT* while the cluster is running. This is because they affect the aggregate failover time, which is only reported to the CVM stack on cluster startup. You also cannot change the quorum configuration while `SG-CFS-pkg` is running.

Previewing the Effect of Cluster Changes

Many variables affect package placement, including the availability of cluster nodes; the availability of networks and other resources on those nodes; failover and failback policies; and package weights, dependencies, and priorities, if you have configured them. You can preview the effect on packages of certain actions or events before they actually occur.

For example, you might want to check to see if the packages are placed as you expect when the cluster first comes up; or preview what happens to the packages running on a given node if the node halts, or if the node is then restarted; or you might want to see the effect on other packages if a currently disabled package has package switching enabled, or if a package halts and cannot restart because none of the nodes on its *node_list* is available.

Serviceguard provides two ways to do this: you can use the preview mode of Serviceguard commands, or you can use the `cmeval (1m)` command to simulate different cluster states.

Alternatively, you might want to model changes to the cluster as a whole; `cmeval` allows you to do this; see "Using cmeval" (page 280).

What You Can Preview

You can preview any of the following, or all of them simultaneously:

- Cluster bring-up (`cmruncl`)
- Cluster node state changes (`cmrunnode`, `cmhaltnode`)
- Package state changes (`cmrunpkg`, `cmhaltpkg`)
- Package movement from one node to another
- Package switching changes (`cmmodpkg -e`)

- Availability of package subnets, EMS resources, and storage
- Changes in package priority, node order, dependency, failover and failback policy, node capacity and package weight

Using Preview mode for Commands and in Serviceguard Manager

The following commands support the `-t` option, which allows you to run the command in preview mode:

- `cmhaltnode [-t] [-f] <node_name>`
- `cmrunnode [-t] <node_name>`
- `cmhaltpkg [-t] <package_name>`
- `cmrunpkg [-t] [-n node_name] <package_name>`
- `cmmodpkg { -e [-t] | -d } [-n node_name] <package_name>`
- `cmruncl -v [-t]`

NOTE: You cannot use the `-t` option with any command operating on a package in maintenance mode; see "Maintaining a Package: Maintenance Mode" (page 273).

For more information about these commands, see their respective manpages. You can also perform these preview functions in Serviceguard Manager: check the `Preview [...]` box for the action in question.

When you use the `-t` option, the command, rather than executing as usual, predicts the results that *would* occur, sending a summary to `$stdout`. For example, assume that `pkg1` is a high-priority package whose primary node is `node1`, and which depends on `pkg2` and `pkg3` to run on the same node. These are lower-priority packages which are currently running on `node2`. `pkg1` is down and disabled, and you want to see the effect of enabling it:

```
cmmodpkg -e -t pkg1
```

You will see output something like this:

```
package:pkg3|node:node2|action:failing
package:pkg2|node:node2|action:failing
package:pkg2|node:node1|action:starting
package:pkg3|node:node1|action:starting
package:pkg1|node:node1|action:starting
cmmodpkg: Command preview completed successfully
```

This shows that `pkg1`, when enabled, will "drag" `pkg2` and `pkg3` to its primary node, `node1`. It can do this because of its higher priority; see "Dragging Rules for Simple Dependencies" (page 130). Running the preview confirms that all three packages will successfully start on `node2` (assuming conditions do not change between now and when you actually enable `pkg1`, and there are no failures in the run scripts).

NOTE: The preview cannot predict run and halt script failures.

For more information about package dependencies and priorities, see "About Package Dependencies" (page 128).

Using cmeval

You can use `cmeval` to evaluate the effect of cluster changes on Serviceguard packages. You can also use it simply to preview changes you are considering making to the cluster as a whole.

You can use `cmeval` safely in a production environment; it does not affect the state of the cluster or packages. Unlike command preview mode (the `-t` discussed above) `cmeval` does not require you to be logged in to the cluster being evaluated, and in fact that cluster does not have to be

running, though it must use the same Serviceguard release and patch version as the system on which you run cmeval.

Use cmeval rather than command preview mode when you want to see more than the effect of a single command, and especially when you want to see the results of large-scale changes, or changes that may interact in complex ways, such as changes to package priorities, node order, dependencies and so on.

Using cmeval involves three major steps:

1. Use cmviewcl -v -f line to write the current cluster configuration out to a file.
2. Edit the file to include the events or changes you want to preview
3. Using the file from Step 2 as input, run cmeval to preview the results of the changes.

For example, assume that pkg1 is a high-priority package whose primary node is node1, and which depends on pkg2 and pkg3 to be running on the same node. These lower-priority-packages are currently running on node2. pkg1 is down and disabled, and you want to see the effect of enabling it.

In the output of cmviewcl -v -f line, you would find the line package:pkg1|autorun=disabled and change it to package:pkg1|autorun=enabled. You should also make sure that the nodes the package is configured to run on are shown as available; for example: package:pkg1|node:node1|available=yes. Then save the file (for example as newstate.in) and run cmeval:

```
cmeval -v newstate.in
```

You would see output something like this:

```
package:pkg3|node:node2|action:failing
package:pkg2|node:node2|action:failing
package:pkg2|node:node1|action:starting
package:pkg3|node:node1|action:starting
package:pkg1|node:node1|action:starting
```

This shows that pkg1, when enabled, will "drag" pkg2 and pkg3 to its primary node, node1. It can do this because of its higher priority; see "Dragging Rules for Simple Dependencies" (page 130). Running cmeval confirms that all three packages will successfully start on node2 (assuming conditions do not change between now and when you actually enable pkg1, and there are no failures in the run scripts.)

NOTE: cmeval cannot predict run and halt script failures.

This is a simple example; you can use cmeval for much more complex scenarios; see "What You Can Preview" (page 279).

IMPORTANT: For detailed information and examples, see the cmeval (1m) manpage.

Updating the Cluster Lock Configuration

Use the procedures that follow whenever you need to change the device file names of the cluster lock physical volumes. You may need to do this because you are changing the device itself (the disk or LUN), or for some other reason — for example, in order to migrate cluster nodes to the agile addressing scheme available as of HP-UX 11i v3 (see "About Device File Names (Device Special Files)" (page 77)), or to migrate cluster DSFs to cDSFs; see "About Cluster-wide Device Special Files (cDSFs)" (page 99).

Updating the Cluster Lock Disk Configuration Online

You can change the device file names (DSFs) of the cluster lock physical volumes (that is, the values of the *FIRST_CLUSTER_LOCK_PV* and *SECOND_CLUSTER_LOCK_PV* parameters in the cluster configuration file) without bringing down the cluster. Proceed as follows.

IMPORTANT: See "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

1. In the cluster configuration file, modify the values of *FIRST_CLUSTER_LOCK_PV* and *SECOND_CLUSTER_LOCK_PV* for each node.
2. Run `cmcheckconf` to check the configuration.
3. Run `cmapplyconf` to apply the configuration.

For information about replacing the physical disk, see "Replacing a Lock Disk" (page 311).

Updating the Cluster Lock LUN Configuration Online

You can change the lock LUN configuration while the cluster is running. Proceed as follows.

IMPORTANT: See "What Happens when You Change the Quorum Configuration Online" (page 47) for important information.

1. In the cluster configuration file, modify the value of *CLUSTER_LOCK_LUN* for each node.
2. Run `cmcheckconf` to check the configuration.
3. Run `cmapplyconf` to apply the configuration.

For information about replacing the physical device, see "Replacing a Lock LUN" (page 312).

Reconfiguring a Halted Cluster

You can make a permanent change in the cluster configuration when the cluster is halted. This procedure *must* be used for changes marked "Cluster must not be running" in Table 13 (page 278), 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 Manager'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, or by entering `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 Manager's `Apply` button, or enter `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 Manager's `Run Cluster` command, or `cmruncl` on the command line.

Reconfiguring a Running Cluster

This section provides instructions for changing the cluster configuration while the cluster is up and running. Note the following restrictions:

- 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.
- 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.

Adding Nodes to the Cluster While the Cluster is Running

You can use Serviceguard Manager to add nodes to a running cluster, or use Serviceguard commands as in the example below.

In 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 cluster1 temp.ascii`

2. Specify a new set of nodes to be configured and generate a template of the new configuration. Specify the node name (39 bytes or less) without its full domain name; for example, `ftsys8` rather than `ftsys8.cup.hp.com`. Enter a command such as the following (all on one line):

   `cmquerycl -C clconfig.ascii -c cluster1 -n ftsys8 -n ftsys9 -n ftsys10`

3. Open `clconfig.ascii` in an editor and check that the information about the new node is what you want.
4. Verify the new configuration:

   `cmcheckconf -C clconfig.ascii`

5. Apply the changes to the configuration and distribute the new binary configuration file to all cluster nodes:

   `cmapplyconf -C clconfig.ascii`

Use `cmrunnode` to start the new node, and, if you so decide, 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: Before you can add a node to a running cluster that uses Veritas CVM, the node must already be connected to the disk devices for all CVM disk groups. The disk groups will be available for import when the node joins the cluster.

Removing Nodes from the Cluster while the Cluster Is Running

You can use Serviceguard Manager to delete nodes, or Serviceguard commands as shown below. The following restrictions apply:

- The node must be halted. See "Removing Nodes from Participation in a Running Cluster" (page 266).
- If the node you want 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.

Use the following procedure to delete a node with HP-UX commands. In this example, nodes `ftsys8`, `ftsys9` and `ftsys10` are already configured in a running cluster named `cluster1`, and you are deleting node `ftsys10`.

NOTE: If you want to remove a node from the cluster, run the `cmapplyconf` command from another node in the same cluster. If you try to issue the command on the node you want removed, you will get an error message.

---

1. Use the following command to store a current copy of the existing cluster configuration in a temporary file:

   `cmgetconf -c cluster1 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. Halt the node you are going to remove (`ftsys10`in this example):

   `cmhaltnode -f -v ftsys10`

5. Verify the new configuration:

   `cmcheckconf -C clconfig.ascii`

6. From `ftsys8` or `ftsys9`, apply the changes to the configuration and distribute the new binary configuration file to all cluster nodes.:

   `cmapplyconf -C clconfig.ascii`

NOTE: If you are trying to remove an unreachable node on which many packages are configured to run (especially if the 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.

Changing the Cluster Networking Configuration while the Cluster Is Running

What You Can Do

Online operations you can perform include:

- Add a network interface with its *`HEARTBEAT_IP`* or *`STATIONARY_IP`*.
- Add a standby interface.
- Delete a network interface with its *`HEARTBEAT_IP`* or *`STATIONARY_IP`*.
- Delete a standby interface.
- Change a *`HEARTBEAT_IP`* or *`STATIONARY_IP`* interface from IPv4 to IPv6, or *vice versa*.
- Change the designation of an existing interface from *`HEARTBEAT_IP`* to *`STATIONARY_IP`*, or *vice versa*.
- Change the *`NETWORK_POLLING_INTERVAL`*.
- Change the *`NETWORK_FAILURE_DETECTION`* parameter.
- Change the *`NETWORK_AUTO_FAILBACK`* parameter.
- Change IP Monitor parameters: *`SUBNET`*, *`IP_MONITOR`*, *`POLLING TARGET`*; see the entries for these parameters under "Cluster Configuration Parameters " (page 105) for more information.
- A combination of any of these in one transaction (`cmapplyconf`), given the restrictions below.

What You Must Keep in Mind

The following restrictions apply:

- You must not change the configuration of all heartbeats at one time, or change or delete the only configured heartbeat.

  At least one working heartbeat, preferably with a standby, must remain unchanged.

- In a CVM configuration, you can add and delete only data LANs and IP addresses.

  You cannot change the heartbeat configuration while a cluster that uses CVM is running.

- You cannot add interfaces or modify their characteristics unless those interfaces, and all other interfaces in the cluster configuration, are healthy.

  There must be no bad NICs or non-functional or locally switched subnets in the configuration, unless you are deleting those components in the same operation.

- You cannot change the designation of an existing interface from *HEARTBEAT_IP* to *STATIONARY_IP*, or *vice versa,* without also making the same change to all peer network interfaces on the same subnet on all other nodes in the cluster.

  Similarly, you cannot change an interface from IPv4 to IPv6 without also making the same change to all peer network interfaces on the same subnet on all other nodes in the cluster

- You cannot change the designation of an interface from *STATIONARY_IP* to *HEARTBEAT_IP* unless the subnet is common to all nodes.

  Remember that the *HEARTBEAT_IP* must on the same subnet on all nodes, except in cross-subnet configurations; see "Cross-Subnet Configurations" (page 29).

- You cannot delete a primary interface without also deleting any standby interfaces, unless the standby is being used by another primary interface that is not being deleted.
- You cannot delete a subnet or IP address from a node while a package that uses it (as a *monitored_subnet*, *ip_subnet*, or *ip_address*) is configured to run on that node.

  See the package networking parameter descriptions (page 229) for more information.

- You cannot change the IP configuration of an interface (NIC) used by the cluster in a single transaction (`cmapplyconf`).

  You must first delete the NIC from the cluster configuration, then reconfigure the NIC (using `ifconfig (1m)`, for example), then add the NIC back into the cluster.

  Examples of when you must do this include:

  - moving a NIC from one subnet to another
  - adding an IP address to a NIC
  - removing an IP address from a NIC

△ **CAUTION:** Do not add IP addresses to network interfaces that are configured into the Serviceguard cluster, unless those IP addresses themselves will be immediately configured into the cluster as stationary IP addresses. If you configure any address other than a stationary IP address on a Serviceguard network interface, it could collide with a relocatable package address assigned by Serviceguard.

Some sample procedures follow.

Example: Adding a Heartbeat LAN

Suppose that a subnet `15.13.170.0` is shared by nodes `ftsys9` and `ftsys10` in a two-node cluster `cluster1`, and you want to add it to the cluster configuration as a heartbeat subnet. Proceed as follows.

1. Run `cmquerycl` to get a cluster configuration template file that includes networking information for interfaces that are available to be added to the cluster configuration:

   `cmquerycl -c cluster1 -C clconfig.ascii`

   NOTE: As of Serviceguard A.11.18, `cmquerycl -c` produces output that includes commented-out entries for interfaces that are not currently part of the cluster configuration, but are available.

   The networking portion of the resulting `clconfig.ascii` file looks something like this:

```
NODE_NAME               ftsys9
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.18
 #NETWORK_INTERFACE     lan0
    #STATIONARY_IP      15.13.170.18
 NETWORK_INTERFACE      lan3
# Possible standby Network Interfaces for lan1, lan0: lan2.
NODE_NAME               ftsys10
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.19
 #NETWORK_INTERFACE     lan0
  # STATIONARY_IP       15.13.170.19
 NETWORK_INTERFACE      lan3
# Possible standby Network Interfaces for lan0, lan1: lan2
```

2. Edit the file to uncomment the entries for the subnet that is being added `lan0` in this example), and change `STATIONARY_IP` to `HEARTBEAT_IP`:

```
NODE_NAME               ftsys9
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.18
 NETWORK_INTERFACE      lan0
    HEARTBEAT_IP        15.13.170.18
 NETWORK_INTERFACE      lan3
# Possible standby Network Interfaces for lan1, lan0: lan2.
NODE_NAME               ftsys10
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.19
 NETWORK_INTERFACE      lan0
  HEARTBEAT_IP          15.13.170.19
 NETWORK_INTERFACE      lan3
# Possible standby Network Interfaces for lan0, lan1: lan2
```

3. Verify the new configuration:

   `cmcheckconf -C clconfig.ascii`

4. Apply the changes to the configuration and distribute the new binary configuration file to all cluster nodes:

   `cmapplyconf -C clconfig.ascii`

If you were configuring the subnet for data instead, and wanted to add it to a package configuration, you would now need to:

1. Halt the package
2. Add the new networking information to the package configuration file
3. In the case of a legacy package, add the new networking information to the package control script if necessary
4. Apply the new package configuration, and redistribute the control script if necessary.

For more information, see "Reconfiguring a Package on a Running Cluster " (page 297).

Example: Deleting a Subnet Used by a Package

In this example, we are deleting subnet 15.13.170.0 (`lan0`). This will also mean deleting `lan3`, which is a standby for `lan0` and not shared by any other primary LAN. Proceed as follows.

1. Halt any package that uses this subnet and delete the corresponding networking information: *`monitored_subnet`*, *`ip_subnet`*, *`ip_address`* (page 229).

   See "Reconfiguring a Package on a Running Cluster " (page 297) for more information.

2. Run `cmquerycl` to get the cluster configuration file:

   `cmquerycl -c cluster1 -C clconfig.ascii`

3. Comment out the network interfaces `lan0` and `lan3` and their network interfaces, if any, on all affected nodes. The networking portion of the

```
NODE_NAME               ftsys9
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.18
 # NETWORK_INTERFACE    lan0
 # STATIONARY_IP        15.13.170.18
 # NETWORK_INTERFACE    lan3
# Possible standby Network Interfaces for lan1, lan0: lan2.
NODE_NAME               ftsys10
 NETWORK_INTERFACE      lan1
    HEARTBEAT_IP        192.3.17.19
 # NETWORK_INTERFACE    lan0
 # STATIONARY_IP        15.13.170.19
 # NETWORK_INTERFACE    lan3
# Possible standby Network Interfaces for lan0, lan1: lan2
```

4. Verify the new configuration:

   `cmcheckconf -C clconfig.ascii`

5. Apply the changes to the configuration and distribute the new binary configuration file to all cluster nodes:

   `cmapplyconf -C clconfig.ascii`

Removing a LAN or VLAN Interface from a Node

You must remove a LAN or VLAN interface from the cluster configuration before removing the interface from the system.

On an HP-UX 11i v3 system, you can then remove the interface without shutting down the node. Follow these steps on the affected node:

NOTE: This can be done on a running system only on HP-UX 11i v3. You must shut down an HP-UX 11i v2 system before removing the interface.

---

1. If you are not sure whether or not a physical interface (NIC) is part of the cluster configuration, run `olrad -C` with the affected I/O slot ID as argument. If the NIC is part of the cluster configuration, you'll see a warning message telling you to remove it from the configuration before you proceed. See the `olrad(1M)` manpage for more information about `olrad`.
2. Use the `cmgetconf` command to store a copy of the cluster's existing cluster configuration in a temporary file. For example:

   `cmgetconf clconfig.ascii`

3. Edit `clconfig.ascii` and delete the line(s) specifying the NIC name and its IP address(es) (if any) from the configuration.
4. Run `cmcheckconf` to verify the new configuration.
5. Run `cmapplyconf` to apply the changes to the configuration and distribute the new configuration file to all the cluster nodes.

6. Run `olrad -d` to remove the NIC.

See also "Replacing LAN or Fibre Channel Cards" (page 314).

Changing the LVM Configuration while the Cluster is Running

You can do this in Serviceguard Manager, or use HP-UX commands as in the example that follows.

NOTE: If you are removing a volume group from the cluster configuration, make sure that you also modify any package that activates and deactivates this volume group. In addition, you should use the LVM `vgexport` command on the removed volume group; do this on each node that will no longer be using the volume group.

From the LVM's cluster, follow these steps:

1. Use the `cmgetconf` command to store a copy of the cluster's existing cluster configuration in a temporary file. For example: **`cmgetconf clconfig.ascii`**
2. Edit the file `clconfig.ascii` to add or delete volume groups.
3. Use the `cmcheckconf` command to verify the new configuration.
4. Use the `cmapplyconf` command to apply the changes to the configuration and distribute the new binary configuration file to all cluster nodes.

NOTE: If the volume group that you are deleting from the cluster 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 configuration file or control script.

Changing the VxVM or CVM Storage Configuration

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CVM and CFS: http://www.hp.com/go/hpux-serviceguard-docs.

You can add VxVM and CVM disk groups to the cluster configuration while the cluster is running. The cluster *must* be running before you can add new CVM disk groups; for instructions, see "Adding Disk Groups to the Package Configuration " (page 210).

Similarly, you can delete VxVM or CVM disk groups provided they are not being used by a cluster node at the time.

△ **CAUTION:** Serviceguard manages the Veritas processes, specifically `gab` and `LLT`. This means that you should never use administration commands such as `gabconfig`, `llthosts`, and `lltconfig` to administer a cluster. It is safe to use the read-only variants of these commands, such as `gabconfig -a`. But a Veritas administrative command could potentially crash nodes or the entire cluster.

NOTE: If you are removing a disk group from the cluster configuration, make sure that you also modify or delete any package configuration file (or legacy package control script) that imports and deports this disk group. Be sure to remove the disk group from the configuration of any package that used it, as well as the corresponding *`dependency_`* parameters.

Changing `MAX_CONFIGURED_PACKAGES`

As of Serviceguard A.11.17, you can change *`MAX_CONFIGURED_PACKAGES`* while the cluster is running. The default for *`MAX_CONFIGURED_PACKAGES`* is the maximum number allowed in the cluster. You can use Serviceguard Manager to change *`MAX_CONFIGURED_PACKAGES`*, or Serviceguard commands as shown below.

Use `cmgetconf` to obtain a current copy of the cluster's existing configuration; for example:

```
cmgetconf -c <cluster_name> clconfig.ascii
```

Edit the `clconfig.ascii` file to include the new value for *MAX_CONFIGURED_PACKAGES*. Then use the `cmcheckconf` command to verify the new configuration. Using the `-k` or `-K` option can significantly reduce the response time.

Use `cmapplyconf` to apply the changes to the configuration and send the new configuration file to all cluster nodes. Using `-k` or `-K` can significantly reduce the response time.

Configuring a Legacy Package

IMPORTANT: You can still create a new legacy package. If you are using a Serviceguard Toolkit such as Serviceguard NFS Toolkit, consult the documentation for that product.

Otherwise, use this section to maintain and rework existing legacy packages rather than to create new ones. The method described in "Configuring Packages and Their Services " (page 216), is simpler and more efficient for creating new packages, allowing packages to be built from smaller modules, and eliminating the separate package control script and the need to distribute it manually.

If you decide to convert a legacy package to a modular package, see "Migrating a Legacy Package to a Modular Package" (page 297). Do not attempt to convert Serviceguard Toolkit packages.

Creating or modifying a legacy package requires the following broad steps:

1. Generate the package configuration file
2. Edit the package configuration file
3. Generate the package control script
4. Edit the package control script
5. Distribute the control script to the cluster nodes
6. Apply the package configuration file

Each of these tasks is described in the subsections that follow.

Creating the Legacy 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 legacy package and its control script in Serviceguard Manager; use the Help for detailed instructions. Otherwise, use the following procedure to create a legacy package.

1. 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 like.

2. Generate a package configuration file for each package, for example:

   ```
   cmmakepkg -p /etc/cmcluster/pkg1/pkg1.conf
   ```

   You can use any file name you like for the configuration file.

3. Edit each configuration file to specify package name, prioritized list of nodes (with 39 bytes or less in the name), the location of the control script, and failover parameters for each package. Include the data you recorded on the Package Configuration Worksheet.

Configuring a Package in Stages

It is a good idea to configure failover packages in stages, as follows:

1. Configure volume groups and mount points only.
2. Distribute the control script to all nodes.

3. Apply the configuration.
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.

Editing the Package Configuration File

Edit the file you generated with `cmmakepkg`. Use the bullet points that follow as a checklist.

NOTE: HP strongly recommends that you never edit the package configuration file of a legacy CVM/CFS multi-node or system multi-node package, although Serviceguard does not prohibit it. Create `SG-CFS-pkg` by issuing the `cmapplyconf` command. Create and modify the legacy CFS packages `SG-CFS-DG-id#` and `SG-CFS-MP-id#` using `cfs` commands.

For the CVM/CFS modular style packages, you cannot use the `cfs` commands. Instead, you must edit the modular CFS package parameters in the package configuration files.

- *PACKAGE_TYPE*. Enter the package type; see "Types of Package: Failover, Multi-Node, System Multi-Node" (page 217) and *package_type* (page 223).

  NOTE: For modular packages, the default form for parameter names in the package configuration file is lower case; for legacy packages the default is upper case. There are no compatibility issues; Serviceguard is case-insensitive as far as the parameter names are concerned.

  Because this section is intended to be used primarily when you are reconfiguring an existing legacy package, we are using the legacy parameter names (in upper case) for sake of continuity. But if you generate the configuration file using `cmmakepkg` or `cmgetconf`, you will see the parameter names as they appear in modular packages; see the notes below and the "Package Parameter Explanations" (page 222) for details of the name changes.

- *FAILOVER_POLICY*. For failover packages, enter the *failover_policy* (page 226).
- *FAILBACK_POLICY*. For failover packages, enter the *failback_policy* (page 227).
- *NODE_NAME*. Enter the node or nodes on which the package can run; see *node_name* (page 223).
- *AUTO_RUN*. Configure the package to start up automatically or manually; see *auto_run* (page 224).
- *LOCAL_LAN_FAILOVER_ALLOWED*. Enter the policy for *local_lan_failover_allowed* (page 229).
- *NODE_FAIL_FAST_ENABLED*. Enter the policy for *node_fail_fast_enabled* (page 224).
- *RUN_SCRIPT* and *HALT_SCRIPT*. Specify the pathname of the package control script (described in the next section). No default is provided. Permissions on the file and directory should be set to `rwxr-xr-x` or `r-xr-xr-x` (`755` or `555`).

  (Script timeouts): Enter the *run_script_timeout* (page 224) and *halt_script_timeout* (page 225).

  SCRIPT_LOG_FILE. (optional). Specify the full pathname of the file where the *RUN_SCRIPT* and *HALT_SCRIPT* will log messages. If you do not specify a path, Serviceguard will create a file with ".log" appended to each script path, and put the messages in that file.

- Enter the *SUBNET* or *MONITORED_SUBNET*s that are to be monitored for this package. They can be IPv4 or an IPv6 subnets, but must not be link-local subnets (link-local package IPs are not allowed).

IMPORTANT: Each subnet specified here must already be specified in the cluster configuration file via the *NETWORK_INTERFACE* parameter and either the *HEARTBEAT_IP* or *STATIONARY_IP* parameter. See "Cluster Configuration Parameters " (page 105) for more information.

See also "Stationary and Relocatable IP Addresses " (page 64) and *monitored_subnet* (page 229).

IMPORTANT: For cross-subnet configurations, see "Configuring Cross-Subnet Failover" (page 295).

- If your package runs services, enter the *SERVICE_NAME* (page 232) and values for *SERVICE_FAIL_FAST_ENABLED* (page 233) and *SERVICE_HALT_TIMEOUT*(page 233). Enter a group of these three for each service.

IMPORTANT: Note that the rules for valid *SERVICE_NAME*s are more restrictive as of A.11.18.

- To configure monitoring for a registered resource, enter values for the following parameters.
  - *RESOURCE_NAME*
  - *RESOURCE_POLLING_INTERVAL*
  - *RESOURCE_UP_VALUE*
  - *RESOURCE_START*

  For more information, see "Parameters for Configuring EMS Resources" (page 128), and the *resource_* parameter descriptions (page 233).

NOTE: For legacy packages, DEFERRED resources must be specified in the package control script.

- *ACCESS_CONTROL_POLICY*. You can grant a non-root user PACKAGE_ADMIN privileges for this package.

  See the entries for *user_name*, *user_host*, and *user_role* (page 240), and "Controlling Access to the Cluster" (page 183), for more information.

- If the package will depend on another package, enter values for *DEPENDENCY_NAME*, *DEPENDENCY_CONDITION*, and *DEPENDENCY_LOCATION*.

  For more information, see the corresponding parameter descriptions (page 227), and "About Package Dependencies" (page 128).

Creating the Package Control Script

For legacy packages, 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, HP-UX commands, or a combination of both, to create or modify the package control script.

Each package must have a separate control script, which must be executable.

For security reasons, the control script 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 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.

IMPORTANT: Serviceguard automatically creates the necessary control scripts when you create the multi-node or system multi-node packages for CVM/CFS (version 4.1 and later). HP strongly recommends that you never edit the configuration or control script files for these packages, although Serviceguard does not prevent it. For CFS, create and modify the information using `cfs` administration commands only. See "Creating a Storage Infrastructure with Veritas Cluster File System (CFS)" (page 190). For CVM without CFS, see "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208)

Use `cmmakepkg` to create the control script, then edit the control script. Use the following procedure to create the template for the sample failover package `pkg1`.

First, generate a control script template, for example:

```
cmmakepkg -s /etc/cmcluster/pkg1/pkg1.sh
```

Next, customize the script; see "Customizing the Package Control Script ".

Customizing the Package Control Script

You need to customize as follows. See the entries for the corresponding modular-package parameters under "Package Parameter Explanations" (page 222) for more discussion.

- 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 file systems, if desired. Do not use the *VXVM_DG[]* or *CVM_DG[]* parameters for LVM volume groups.
- If you are using CVM disk group for raw access (without CFS), 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. Configure *vxvm_dg_retry* (page 236) if necessary. Do not use the *CVM_DG[]* or *VG[]* parameters for VxVM disk groups without CVM, and do not specify an activation command.

  Do not include CFS-based disk groups in the package control script; they are activated by the CFS multi-node packages before standard packages are started.
- If you are using mirrored VxVM disks, specify the mirror recovery option *VXVOL*.
- Add the names of logical volumes and the file system 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 file systems, if desired.
- Specify the file system mount and unmount retry 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.

- If your package will use relocatable IP addresses, define IP subnet and IP address pairs. IPv4 or IPv6 addresses are allowed.

△ **CAUTION:** HP recommends that the subnet(s) be specified in the cluster configuration file via the *NETWORK_INTERFACE* parameter and either the *HEARTBEAT_IP* or *STATIONARY_IP* parameter; see "Cluster Configuration Parameters " (page 105).

If you do not follow this recommendation, the package may not behave as you expect; see *ip_subnet* (page 231), *ip_address* (page 232), and "Stationary and Relocatable IP Addresses " (page 64) for complete information.

- Add service name(s).
- Add service command(s)
- Add a service restart parameter, if you so decide.

  For more information about services, see the discussion of the *service_* parameters (page 232).
- Specify whether or not to kill processes accessing raw devices; see the comments in the file under `RAW DEVICES` for more information.

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. Enter these commands in the `CUSTOMER DEFINED FUNCTIONS` area of the script.

If your package needs to run short-lived processes, such as commands to initialize or halt a packaged application, you can also run these from the `CUSTOMER DEFINED FUNCTIONS`.

You can also use the `CUSTOMER DEFINED FUNCTIONS` to determine why a package has shut down; see "Determining Why a Package Has Shut Down" (page 145).

An example of this portion of the script follows, showing the `date` and `echo` commands logging starts and halts of the package to a 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. These commands must not interact with the package itself.

If a Serviceguard command interacts with another package, be careful to avoid command loops. 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/XP and EVA, 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 checks the configuration you create and reports any errors.

For legacy packages, you can do this in Serviceguard Manager: click `Check` to verify the package configuration you have done under any package configuration tab, or to check changes you have made to the control script. Click `Apply` to verify the package as a whole. See the local Help for more details.

If you are using the command line, use the following command to verify the content of the package configuration you have created:

```
cmcheckconf -v -P /etc/cmcluster/pkg1/pkg1.conf
```

Errors are displayed on the standard output. If necessary, edit the file to correct any errors, then run the command again until it completes without errors.

The following items are checked (whether you use Serviceguard Manager or `cmcheckconf` command):

- 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.
- If a dependency is configured, the dependency package must already be configured in the cluster.

Distributing the Configuration

You can use Serviceguard Manager or HP-UX commands to distribute the binary cluster configuration file among the nodes of the cluster.

DSAU (Distributed Systems Administration Utilities) can help you streamline your distribution; see "What are the Distributed Systems Administration Utilities?" (page 24).

Distributing the Configuration And Control Script with Serviceguard Manager

When you have finished creating a legacy package in Serviceguard Manager, click `Apply Configuration`. If the package control script has no errors, it is converted to a binary file and distributed to the cluster nodes.

Copying Package Control Scripts with HP-UX commands

IMPORTANT: In a cross-subnet configuration, you cannot use the same package control script on all nodes if the package uses relocatable IP addresses. See "Configuring Cross-Subnet Failover" (page 295).

Use HP-UX commands to copy legacy package control scripts from the node where you created the files, 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 run the `rcp` command to copy the package control script to `ftsys10` (all on one line):

```
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 you created the cluster and package configuration files:

- Verify that the configuration file is correct. Use the following command (all on one line):

  ```
  cmcheckconf -C /etc/cmcluster/cmcl.conf -P
  /etc/cmcluster/pkg1/pkg1.conf
  ```

- 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 (all on one line):

  ```
  cmapplyconf -v -C /etc/cmcluster/cmcl.conf -P
  /etc/cmcluster/pkg1/pkg1.conf
  ```

- 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: You must use `cmcheckconf` and `cmapplyconf` again any time you make changes to the cluster and package configuration files.

Configuring Cross-Subnet Failover

To configure a legacy package to fail over across subnets (see "Cross-Subnet Configurations" (page 29)), you need to do some additional configuration.

NOTE: You cannot use Serviceguard Manager to configure cross-subnet packages.

Suppose that you want to configure a package, `pkg1`, so that it can fail over among all the nodes in a cluster comprising `NodeA`, `NodeB`, `NodeC`, and `NodeD`.

`NodeA` and `NodeB` use subnet `15.244.65.0`, which is not used by `NodeC` and `NodeD`; and `NodeC` and `NodeD` use subnet `15.244.56.0`, which is not used by `NodeA` and `NodeB`. (See "Obtaining Cross-Subnet Information" (page 181) for sample `cmquerycl` output).

Configuring node_name

First you need to make sure that pkg1 will fail over to a node on another subnet only if it has to. For example, if it is running on NodeA and needs to fail over, you want it to try NodeB, on the same subnet, before incurring the cross-subnet overhead of failing over to NodeC or NodeD.

NOTE: If you are using a site-aware disaster-tolerant cluster, which requires Metrocluster (additional HP software), you can use the *SITE* to accomplish this. See the description of that parameter under "Cluster Configuration Parameters " (page 105).

Assuming nodeA is pkg1's primary node (where it normally starts), create *node_name* entries in the package configuration file as follows:

```
node_name nodeA
node_name nodeB
node_name nodeC
node_name nodeD
```

Configuring monitored_subnet_access

In order to monitor subnet 15.244.65.0 or 15.244.56.0, you would configure *monitored_subnet* and *monitored_subnet_access* in pkg1's package configuration file as follows:

```
monitored_subnet 15.244.65.0
monitored_subnet_access PARTIAL
monitored_subnet 15.244.56.0
monitored_subnet_access PARTIAL
```

NOTE: Configuring *monitored_subnet_access* as FULL (or not configuring *monitored_subnet_access*) for either of these subnets will cause the package configuration to fail, because neither subnet is available on all the nodes.

Creating Subnet-Specific Package Control Scripts

Now you need to create control scripts to run the package on the four nodes.

IMPORTANT: In a cross-subnet configuration, you cannot share a single package control script among nodes on different subnets if you are using relocatable IP addresses. In this case you will need to create a separate control script to be used by the nodes on each subnet.

In our example, you would create two copies of pkg1's package control script, add entries to customize it for subnet 15.244.65.0 or 15.244.56.0, and copy one of the resulting scripts to each node, as follows.

Control-script entries for nodeA and nodeB

```
IP[0] = 15.244.65.82
SUBNET[0] 15.244.65.0
IP[1] = 15.244.65.83
SUBNET[1] 15.244.65.0
```

Control-script entries for nodeC and nodeD

```
IP[0] = 15.244.56.100
SUBNET[0] = 15.244.56.0
IP[1] = 15.244.56.101
```

```
SUBNET[1] = 15.244.56.0
```

Reconfiguring a Package

You reconfigure a package in much the same way as you originally configured it; for modular packages, see Chapter 6: "Configuring Packages and Their Services " (page 216); for older packages, see "Configuring a Legacy Package" (page 289).

The cluster, and the package itself, can be either halted or running during package reconfiguration; see "Reconfiguring a Package on a Running Cluster " (page 297). The types of changes that can be made and the times when they take effect depend on whether the package is running or not; see "Allowable Package States During Reconfiguration " (page 300).

△ **CAUTION:** Be extremely cautious about changing a package's configuration while the package is running.

If you reconfigure a package online (by executing `cmapplyconf` on a package while the package itself is running) it is possible that the package will fail, even if the `cmapplyconf` succeeds, validating the changes with no errors.

For example, if a file system is added to the package while the package is running, `cmapplyconf` does various checks to verify that the file system and its mount point exist. But the actual file system check and mount of the file system can only be done after `cmapplyconf` succeeds; and if one of these tasks fails in a running package, the entire package will fail.

As a rule of thumb, configuration changes which would have prevented a package that was changed offline from starting, will very probably cause the package to fail if the changes are made while the package is running. Be particularly cautious about adding, removing, or changing logical volumes, volume groups, or file systems.

For any change you intend to make, read the information under "Allowable Package States During Reconfiguration " (page 300) carefully, and try out changes on a non-production package before applying them to a running production package.

Migrating a Legacy Package to a Modular Package

The Serviceguard command `cmmigratepkg` automates the process of migrating legacy packages to modular packages as far as possible. Many, but not all, packages can be migrated in this way; for details, see the white paper *Migrating Packages from Legacy Style to Modular Style* at http://www.hp.com/go/hpux-serviceguard-docs.

Do not attempt to convert Serviceguard Toolkit packages.

NOTE: The `cmmigratepkg` command requires Perl version 5.8.3 or higher on the system on which you run the command. It should already be on the system as part of the HP-UX base product.

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. You can do this in Serviceguard Manager (for legacy packages), or use Serviceguard commands.

To modify the package with Serviceguard commands, use the following procedure (`pkg1` is used as an example):

1. Halt the package if necessary:

   ```
   cmhaltpkg pkg1
   ```

△ **CAUTION:** Make sure you read and understand the information and caveats under "Allowable Package States During Reconfiguration " (page 300) before you decide to reconfigure a running package.

2. If it is not already available, obtain a copy of the package's configuration file by using the `cmgetconf` command, specifying the package name.

   ```
   cmgetconf -p pkg1 pkg1.conf
   ```

3. Edit the package configuration file.

⊘ **IMPORTANT:** Restrictions on package names, dependency names, and service names have become more stringent as of A.11.18. Packages that have or contain names that do not conform to the new rules (spelled out under *package_name* (page 222)) will continue to run, but if you reconfigure these packages, you will need to change the names that do not conform; `cmcheckconf` and `cmapplyconf` will enforce the new rules.

4. Verify your changes as follows:

   ```
   cmcheckconf -v -P pkg1.conf
   ```

5. Distribute your changes to all nodes:

   ```
   cmapplyconf -v -P pkg1.ascii
   ```

6. If this is a legacy package, copy the package control script to all nodes that can run the package.

Renaming or Replacing an External Script Used by a Running Package

In most cases, you can rename an *external_script* (page 239) while the package that uses it is running, but you need to be careful; follow the instructions below.

1. Make a copy of the old script, save it with the new name, and edit the copy as needed.
2. Edit the package configuration file to use the new name.
3. Distribute the new script to all nodes that are configured for that package.

   Make sure you place the new script in the correct directory with the proper file modes and ownership.
4. Run `cmcheckconf` to validate the package configuration with the new external script.

△ **CAUTION:** If `cmcheckconf` fails, do not proceed to the next step until you have corrected all the errors.

5. Run `cmapplyconf` on the running package.

   This will stop any resources started by the original script, and then start any resources needed by the new script.
6. You can now safely delete the original external script on all nodes that are configured to run the package.

Reconfiguring a Package on a Halted Cluster

You can also make permanent changes in package configuration while the cluster is not running. Use the same steps as in "Reconfiguring a Package on a Running Cluster " (page 297).

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 *MAX_CONFIGURED_PACKAGES* in the cluster configuration file.

To create the package, follow the steps in the chapter "Configuring Packages and Their Services " (page 216). Use a commands such as the following to verify the configuration of a newly created `pkg1` and distribute the configuration to all nodes in the cluster:

```
cmcheckconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

```
cmapplyconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

If this is a legacy package, remember to copy the control script to the`/etc/cmcluster/pkg1` directory on all nodes that can run the package.

To create the CVM disk group or CFS mount point multi-node packages on systems that support CFS, see "Creating the Disk Group Cluster Packages" (page 194) and "Creating a File System and Mount Point Package" (page 205).

Deleting a Package from a Running Cluster

Serviceguard will not allow you to delete a package if any other package is dependent on it. To check for dependencies, use `cmviewcl -v -l <package>`. System multi-node packages cannot be deleted from a running cluster.

You can use Serviceguard Manager to delete the package.

On the Serviceguard command line, you can (in most cases) delete a package from all cluster nodes by using the `cmdeleteconf` command. To delete one of the Veritas Cluster File System legacy packages, use `cfscluster`, `cfsdgadm`, or `cfsmntadm`. This removes the package information from the binary configuration file on all the nodes in the cluster. The command can only be executed when the package is down; the cluster can be up. To delete modular packages, use `cmhaltpkg` and `cmdeleteconf` commands.

The following example halts the failover 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.

On systems that support CFS, you can remove nodes from a multi-node package configuration using `cfs` commands. All the packages that depend on the multi-node package must be halted on that node.

To remove the legacy CVM disk groups and CFS mount points, follow these steps:

△ **CAUTION:** You must not use the HP-UX `mount` and `umount` commands in a CFS environment; use `cfsmount` or `cfsumount` for legacy CFS packages. For modular packages, you must use `cmcheckconf`, `cmapplyconf`, `cmrunpkg`, `cmmodpkg`, and `cmrunpkg`. Non-CFS commands (for example, `mount -o cluster`, `dbed_chkptmount`, or `sfrac_chkptmount`) could cause conflicts with subsequent operations on the file system or Serviceguard packages, and will not create an appropriate multi-node package, with the result that cluster packages are not aware of file system changes.

1. Remove any dependencies on the package being deleted. Delete *`dependency_`* parameters from the failover application package configuration file, then apply the modified configuration file:

   `cmapplyconf -v -P app1.conf`

2. Unmount the shared file system

   `cfsumount <mount point>`

3. Remove the mount point package from the cluster

   `cfsmntadm delete <mount point>`

   This disassociates the mount point from the cluster. When there is a single volume group associated with the mount point, the disk group package will also be removed

4. Remove the disk group package from the cluster. This disassociates the disk group from the cluster.

   `cfsdgadm delete <disk group>`

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.

When a package service successfully restarts after several attempts, the package manager does not automatically reset the restart count. You can reset the counter online using the **`cmmodpkg -R -s`** command. For example:

`cmmodpkg -R -s myservice pkg1`

The current value of the restart counter is shown in the output of `cmviewcl -v`.

Allowable Package States During Reconfiguration

In many cases, you can make changes to a package's configuration while the package is running. The table that follows shows exceptions — cases in which the package must not be running, or in

which the results might not be what you expect — as well as differences between modular and legacy packages.

CAUTION: Be extremely cautious about changing a package's configuration while the package is running.

If you reconfigure a package online (by executing `cmapplyconf` on a package while the package itself is running) it is possible that the package will fail, even if the `cmapplyconf` succeeds, validating the changes with no errors.

For example, if a file system is added to the package while the package is running, `cmapplyconf` does various checks to verify that the file system and its mount point exist. But the actual file system check and mount of the file system can be done only after `cmapplyconf` succeeds; and if one of these tasks fails in a running package, the entire package will fail.

Another example involves renaming, modifying, or replacing an external script while the package that uses it is running. If the package depends on resources that are managed by the script, the package will fail when you replace the script. See "Renaming or Replacing an External Script Used by a Running Package" (page 298).

As a rule of thumb, configuration changes which would have prevented a package that was changed offline from starting, will very probably cause the package to fail if the changes are made while the package is running. Be particularly cautious about adding, removing, or changing logical volumes, volume groups, or file systems.

For any change you intend to make, read the information in the table carefully, and try out changes on a non-production package before applying them to a running production package.

In general, you have greater scope for online changes to a modular than to a legacy package. In some cases, though, the capability of legacy packages has been upgraded to match that of modular packages as far as possible; these cases are shown in the table. For more information about legacy and modular packages, see Chapter 6 (page 216).

NOTE: If neither *legacy* nor *modular* is called out under "Change to the Package", the "Required Package State" applies to both types of package. Changes that are allowed, but which HP does not recommend, are labeled "should not be running".

IMPORTANT: Actions not listed in the table can be performed for both types of package while the package is running.

In all cases the *cluster* can be running, and packages other than the one being reconfigured can be running. And remember too that you can make changes to package configuration *files* at any time; but do not apply them (using `cmapplyconf` or Serviceguard Manager) to a running package in the cases indicated in the table.

NOTE: All the nodes in the cluster must be powered up and accessible when you make package configuration changes.

Table 14 Types of Changes to Packages

Change to the Package	Required Package State
Delete a package or change package name	Package must not be running. **NOTE:** You cannot delete a package if another package has a dependency on it.
Change package type	Package must not be running.
Add or delete a module: *modular package*	Package can be running.

Table 14 Types of Changes to Packages *(continued)*

Change to the Package	Required Package State
Change run script contents: *legacy package*	Package can be running, but should not be starting. Timing problems may occur if the script is changed while the package is starting.
Change halt script contents: *legacy package*	Package can be running, but should not be halting. Timing problems may occur if the script is changed while the package is halting.
Add or delete a service: *modular package*	Package can be running. Serviceguard treats any change to `service_name` or `service_cmd` as deleting the existing service and adding a new one, meaning that the existing service is halted.
Add or delete a service: *legacy package*	Package must not be running.
Change `service_restart`: *modular package*	Package can be running. Serviceguard will not allow the change if the new value is less than the current restart count. (You can use `cmmodpkg -R<service_name> <package>` to reset the restart count if you need to.)
Change `SERVICE_RESTART`: *legacy package*	Package must not be running.
Add or remove a `SUBNET` (in control script): *legacy package*	Package must not be running. (Also applies to cross-subnet configurations.)
Add or remove an `ip_subnet`: *modular package*	Package can be running. See "`ip_subnet`" (page 231) for important information. Serviceguard will reject the change if you are trying to add an `ip_subnet` that is not configured on all the nodes on the package's `node_name` list.
Add or remove an `ip_address`: *modular package*	Package can be running. See "`ip_subnet`" (page 231) and "`ip_address`" (page 232) for important information. Serviceguard will reject the change if you are trying to add an `ip_address` that cannot be configured on the specified `ip_subnet`, or is on a subnet that is not configured on all the nodes on the package's `node_name` list.
Add or remove an `IP` (in control script): *legacy package*	Package must not be running. (Also applies to cross-subnet configurations.)
Add or delete nodes from package's `ip_subnet_node` list in cross-subnet configurations (page 231)): *modular package*	Package can be running. Serviceguard will reject the change if you are trying to add a node on which the specified `ip_subnet` is not configured.
Add, delete, or change `cluster_interconnect_subnet`	Package must not be running.
Add or remove monitoring for a subnet: `monitored_subnet` for a *modular package* or `SUBNET` (in the package configuration file) for a *legacy package*	Package can be running. Serviceguard will not allow the change if the subnet being added is down, as that would cause the running package to fail.
Change `local_lan_failover_allowed`	Package can be running. Serviceguard will not allow the change if it would cause the package to fail.
Add or remove a resource: *modular package*	Package can be running. Serviceguard will not allow the change if it would cause the package to fail. In addition, Serviceguard will reject the change if the resource is not `UP` within about 30 seconds. Multiple changes that take longer than this should be done

Table 14 Types of Changes to Packages *(continued)*

Change to the Package	Required Package State
	incrementally; resources that are very slow to come up may need to be configured offline. Serviceguard treats a change to *resource_name* as deleting the existing resource and adding a new one, meaning that the existing resource is stopped.
Change *resource_polling_interval*, *resource_start*, *resource_up_value*: *modular package*	Package can be running. Serviceguard will not allow a change to *resource_up_value* if it would cause the package to fail.
Add or remove a resource: *legacy package*	For AUTOMATIC resources, package can be running. See the entry for "Add or remove a resource: *modular package*" for more information. For DEFERRED resources, package must not be running.
Change *RESOURCE_POLLING_INTERVAL* *RESOURCE_UP_VALUE*: *legacy package*	For AUTOMATIC resources, package can be running. For DEFERRED resources, package must not be running. Serviceguard will not allow a change to *RESOURCE_UP_VALUE* if it would cause the package to fail.
Change *RESOURCE_START*: *legacy package*	Package must not be running.
Add a volume group: *modular package*	Package can be running. (Remember that a volume group must already be configured into the cluster, via the *VOLUME_GROUP* parameter, before you can configure it into to a package.)
Add a volume group: *legacy package*	Package must not be running.
Remove a volume group: *modular package*	Package should not be running. **CAUTION:** Removing a volume group may cause problems if file systems, applications or other resources are using it. In particular, if the volume group cannot be deactivated and *kill_processes_accessing_raw_devices* is set to yes, processes accessing the volume group as a raw device will be killed; in addition the CAUTION under "Remove a file system: *modular package*" applies to any file systems using the volume group.
Remove a volume group: *legacy package*	Package must not be running.
Change a file system: *modular package*	Package should not be running (unless you are only changing *fs_umount_opt*). Changing file-system options other than *fs_umount_opt* may cause problems because the file system must be unmounted (using the existing *fs_umount_opt*) and remounted with the new options; the CAUTION under "Remove a file system: *modular package*" applies in this case as well. If *only* *fs_umount_opt* is being changed, the file system will not be unmounted; the new option will take effect when the package is halted or the file system is unmounted for some other reason.
Add a file system: *modular package*	Package can be running.
Add or change a file system: *legacy package*	Package must not be running.
Remove a file system: *modular package*	Package should not be running.

Table 14 Types of Changes to Packages *(continued)*

Change to the Package	Required Package State
	CAUTION: Removing a file system may cause problems if the file system cannot be unmounted because it's in use by a running process. In this case Serviceguard kills the process; this could cause the package to fail.
Remove a file system: *legacy package*	Package must not be running.
Change *concurrent_fsck_operations*, *concurrent_mount_and_umount_operations*, *fs_mount_retry_count*, *fs_umount_retry_count*: *modular package*	Package can be running. These changes in themselves will not cause any file system to be unmounted.
Change *concurrent_fsck_operations*, *concurrent_mount_and_umount_operations*, *fs_mount_retry_count*, *fs_umount_retry_count*: *legacy package*	Package must not be running.
Change *kill_processes_accessing_raw_devices*: *modular package*	Package can be running.
Change *kill_processes_accessing_raw_devices*: *legacy package*	Package must not be running.
Add a *vxvm_dg* or *cvm_dg*: *modular package*	Package can be running.
Remove a *vxvm_dg* or *cvm_dg*: *modular package*	Package should not be running. See CAUTION under "Remove a volume group: *modular package*".
Change *vxvm_dg_retry*: *modular package*	Package can be running.
Add or remove a *VXVM_DG* or *CVM_DG*: *legacy package*	Package must not be running.
Change *VXVM_DG_RETRY*: *legacy package*	Package must not be running.
Add, change, or delete external scripts and pre-scripts: *modular package*	Package can be running but proceed with caution. See "Renaming or Replacing an External Script Used by a Running Package" (page 298). Changes take effect when applied, whether or not the package is running. If you add a script, Serviceguard validates it and then (if there are no errors) runs it when you apply the change. If you delete a script, Serviceguard stops it when you apply the change.
Change *vxvol_cmd*	Package must not be running.
Change *vgchange_cmd*	Package must not be running.
Change *concurrent_vgchange_operations*, *deactivation_retry_count*, *enable_threaded_vgchange*: *modular package*	Package can be running. These changes in themselves will not cause any volume group to be activated or deactivated.
Change *cvm_activation_cmd*	Package must not be running.

Table 14 Types of Changes to Packages *(continued)*

Change to the Package	Required Package State
Add, change, or delete `pev_`: *modular package*	Package can be running.
Change package `auto_run`	Package can be running. See "Choosing Switching and Failover Behavior" (page 127).
Add or delete a configured dependency	Both packages can be either running or halted. Special rules apply to packages in maintenance mode; see "Dependency Rules for a Package in Maintenance Mode or Partial-Startup Maintenance Mode " (page 276). For dependency purposes, a package being reconfigured is considered to be `UP`. This means that if `pkgA` depends on `pkgB`, and `pkgA` is down and `pkgB` is being reconfigured, `pkgA` will run if it becomes eligible to do so, even if `pkgB`'s reconfiguration is not yet complete. HP recommends that you separate package dependency changes from changes that affect resources and services that the newly dependent package will also depend on; reconfigure the resources and services first and apply the changes, then configure the package dependency. For more information see "About Package Dependencies" (page 128).
Add CFS packages	The `SG-CFS-pkg` package must be running, before a modular CFS package can be started. For information on modular CFS packages, see "Creating Modular Disk Group and Mount Point Packages" (page 196). For legacy style packages, before you can add an `SG-CFS-DG-id#` disk group package, the `SG-CFS-pkg` Cluster File System package must be up and running. Before you can add an `SG-MP-id#` mount point package to a node, the `SG-DG-id#` disk group package must be up and running on that node.
Change `cfs_mount_options` for Mount Point for *modular CFS packages*	Package can be running. See "Creating Modular Disk Group and Mount Point Packages" (page 196).
Change `cfs_primary_policy` for Mount Point for *modular CFS packages*	Package can be running. See "Creating Modular Disk Group and Mount Point Packages" (page 196).
Change `ckpt_mount_options` for Checkpoint for *modular CFS packages*	Package can be running. See "Creating Modular Checkpoint and Snapshot Packages for CFS" (page 199).
Change `ckpt_primary_policy` for Checkpoint for *modular CFS packages*	Package can be running. See "Creating Modular Checkpoint and Snapshot Packages for CFS" (page 199).

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information on CVM and CFS support: http://www.hp.com/go/hpux-serviceguard-docs.

Changes that Will Trigger Warnings

Changes to the following will trigger warnings, giving you a chance to cancel, if the change would cause the package to fail.

NOTE: You will not be able to cancel if you use `cmapplyconf -f`.

- Package nodes
- Package dependencies
- Package weights (and also node capacity, defined in the cluster configuration file)

- Package priority
- *auto_run*
- *failback_policy*

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.

Single-Node Operation

In a multi-node cluster, you could have a situation in which all but one node has failed, or you have shut down all but one node, leaving your cluster in single-node operation. This remaining node will probably have applications running on it. As long as the Serviceguard daemon `cmcld` is active, other nodes can rejoin the cluster.

If the Serviceguard daemon fails when in single-node operation, it will leave the single node up and your applications running. (This is different from the loss of the Serviceguard daemon in a multi-node cluster, which halts the node with a TOC, and causes packages to be switched to adoptive nodes.) It is not necessary to halt the single node in this scenario, since the application is still running, and no other node is currently available for package switching.

You should not try to restart Serviceguard, since data corruption might occur if another node were to attempt to start up a new instance of the application that is still running on the single node.

Instead of restarting the cluster, choose an appropriate time to shut down the applications and reboot the node; this will allow Serviceguard to restart the cluster after the reboot.

Disabling Serviceguard

If for some reason you want to disable Serviceguard on a system, you can do so by commenting out the following entries in `/etc/inetd.conf`:

```
hacl-cfg dgram udp wait root /usr/lbin/cmclconfd cmclconfd -p
hacl-cfg stream tcp nowait root /usr/lbin/cmclconfd cmclconfd -c
```

Then force `inetd` to re-read `inetd.conf`:

```
/usr/sbin/inetd -c
```

You can check that this did in fact disable Serviceguard by trying the following command:

```
cmquerycl -n nodename
```

where *nodename* is the name of the local system. If the command fails, you have successfully disabled Serviceguard.

NOTE: You should not disable Serviceguard on a system on which it is actually running. If you are not sure, you can get an indication by means of the command:

```
ps -e | grep cmclconfd
```

If there are `cmclconfd` processes running, it does not mean for certain that Serviceguard is running on this system (`cmclconfd` could simply be handling UDP queries from a Serviceguard cluster on the same subnet) but it does mean you should investigate further before disabling Serviceguard.

Removing Serviceguard from a System

To remove Serviceguard from a node, use the `swremove` command.

CAUTION: Remove the node from the cluster first. If you run the `swremove` command on a server that is still a member of a cluster, it will cause that cluster to halt, and the cluster configuration to be deleted.

To remove Serviceguard:

1. If the node is an active member of a cluster, halt the node.
2. If the node is included in a cluster configuration, remove the node from the configuration.
3. If you are removing Serviceguard from more than one node, issue `swremove` 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

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: from the System Management Homepage (SMH), select the cluster and choose `Administration -> 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 (see "Moving a Failover Package " (page 273)).

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 (see "Moving a Failover Package " (page 273)).
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.
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. On the command line, use `cmviewcl -v`.
4. Reconnect the LAN to the original Primary card, and verify its status. In Serviceguard Manager, check the cluster properties. On the command line, use `cmviewcl -v`.

Monitoring Hardware

Good standard practice in handling a high availability system includes careful fault monitoring so as to prevent failures if possible or at least to react to them swiftly when they occur. 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.

When the proper redundancy has been configured, failures can occur with no external symptoms. Proper monitoring is important. For example, if a Fibre Channel switch in a redundant mass storage configuration fails, LVM will automatically fail over to the alternate path through another Fibre Channel switch. Without monitoring, however, you may not know that the failure has occurred,

since the applications are still running normally. But at this point, there is no redundant path if another failover occurs, so the mass storage configuration is vulnerable.

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.

See the manual *Using High Availability Monitors* at the address given in the preface to this manual.

Using EMS (Event Monitoring Service) 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.

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 ISEE (HP Instant Support Enterprise Edition)

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 ISEE, 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 ISEE should be run on all nodes.

HP ISEE also reports error conditions directly to an HP Response Center, alerting support personnel to the potential problem. HP ISEE 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.

For more information, see the section *Replacing a Bad Disk* in the Logical Volume Management volume of the HP-UX *System Administrator's Guide*, at http://www.hp.com/go/hpux-core-docs.

Replacing a Faulty Array Mechanism

With any HA disk array configured in RAID 1 or RAID 5, refer to the array's documentation for instructions on how to replace a faulty mechanism. After the replacement, the device itself automatically rebuilds the missing data on the new disk. No LVM or VxVM 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 example, 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.

   NOTE: This example assumes you are using legacy DSF naming. Under agile addressing, the physical volume would have a name such as `/dev/disk/disk1`. See "About Device File Names (Device Special Files)" (page 77).

   If you are using cDSFs, the device file would be in the `/dev/rdisk/` directory; for example `/dev/rdisk/disk1`. See "About Cluster-wide Device Special Files (cDSFs)" (page 99).

   If you need to replace a disk under the 11i v3 agile addressing scheme (also used by cDSFs), you may be able to reduce downtime by using the `io_redirect_dsf(1M)` command to reassign the existing DSF to the new device. See the section *Replacing a Bad Disk* in the Logical Volume Management volume of the HP-UX *System Administrator's Guide*, posted at http://www.hp.com/go/hpux-core-docs.

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

You can replace an unusable lock disk while the cluster is running. You can do this without any cluster reconfiguration if you do not change the devicefile name (Device Special File, or DSF); or, if you need to change the DSF, you can do the necessary reconfiguration while the cluster is running.

IMPORTANT: If you need to replace a disk under the HP-UX 11i v3 agile addressing scheme, also used by cDSFs (see "About Device File Names (Device Special Files)" (page 77) and "About Cluster-wide Device Special Files (cDSFs)" (page 99)), and you use the same DSF, you may need to use the `io_redirect_dsf(1M)` command to reassign the existing DSF to the new device, depending on whether the operation changes the WWID of the device. See the section *Replacing a Bad Disk* in the Logical Volume Management volume of the HP-UX *System Administrator's Guide*, posted at http://www.hp.com/go/hpux-core-docs. See also the section on `io_redirect_dsf` at the same address.

If you do not use the existing DSF for the new device, you must change the name of the DSF in the cluster configuration file and re-apply the configuration; see "Updating the Cluster Lock Disk Configuration Online" (page 281). Do this after running `vgcfgrestore` as described below.

CAUTION: Before you start, make sure that all nodes have logged a message in `syslog` saying that the lock disk is corrupt or unusable.

Replace a failed LVM lock disk in the same way as you replace a data disk. If you are using a *dedicated* lock disk (one with no user data on it), then you need to use only one LVM command, for example:

```
vgcfgrestore -n /dev/vg_lock /dev/dsk/c2t3d0
```

Serviceguard checks the lock disk every 75 seconds. After using the `vgcfgrestore` command, review the `syslog` file of an active cluster node for not more than 75 seconds. By this time you should see a message showing that the lock disk is healthy again.

NOTE: If you restore or recreate the volume group for the lock disk and you need to re-create the cluster lock (for example if no `vgcfgbackup` is available), you can run `cmdisklock` to re-create the lock. See the `cmdisklock (1m)` manpage for more information.

Replacing a Lock LUN

You can replace an unusable lock disk while the cluster is running. You can do this without any cluster reconfiguration if you do not change the devicefile name (Device Special File, or DSF); or, if you need to change the DSF, you can do the necessary reconfiguration while the cluster is running.

IMPORTANT: If you need to replace a LUN under the HP-UX 11i v3 agile addressing scheme, also used by cDSFs (see "About Device File Names (Device Special Files)" (page 77) and "About Cluster-wide Device Special Files (cDSFs)" (page 99), and you use the same DSF, you may need to use the `io_redirect_dsf(1M)` command to reassign the existing DSF to the new device, depending on whether the operation changes the WWID of the LUN; see the section on `io_redirect_dsf` in the white paper *The Next Generation Mass Storage Stack* at http://www.hp.com/go/hpux-core-docs.

If you are not able to use the existing DSF for the new device, or you decide not to, you must change the name of the DSF in the cluster configuration file and re-apply the configuration; see "Updating the Cluster Lock LUN Configuration Online" (page 282). Do this after running `vgcfgrestore` as described below.

CAUTION: Before you start, make sure that all nodes have logged a message such as the following in `syslog`:

```
WARNING: Cluster lock LUN /dev/dsk/c0t1d1 is corrupt: bad label. Until
this situation is corrected, a single failure could cause all nodes in
the cluster to crash.
```

Once all nodes have logged this message, use a command such as the following to specify the new cluster lock LUN:

```
cmdisklock reset /dev/dsk/c0t1d1
```

`cmdisklock` checks that the specified device is not in use by LVM, VxVM, ASM, or the file system, and will fail if the device has a label marking it as in use by any of those subsystems. `cmdisklock -f` overrides this check.

CAUTION: You are responsible for determining that the device is not being used by any subsystem on any node connected to the device before using `cmdisklock -f`. If you use `cmdisklock -f` without taking this precaution, you could lose data.

NOTE: `cmdisklock` is needed only when you are repairing or replacing a lock LUN or lock disk; see the `cmdisklock (1m)` manpage for more information.

Serviceguard checks the lock LUN every 75 seconds. After using the `cmdisklock` command, review the `syslog` file of an active cluster node for not more than 75 seconds. By this time you should see a message showing that the lock LUN is healthy again.

Online Hardware Maintenance with In-line SCSI Terminator

In some shared SCSI bus configurations, online 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.

Replacing I/O Cards

Replacing SCSI Host Bus Adapters

After a SCSI Host Bus Adapter (HBA) card failure, you can replace the card using the following steps.

Normally disconnecting any portion of the SCSI bus will leave the SCSI bus in an unterminated state, which will cause I/O errors for other nodes connected to that SCSI bus, so the cluster would need to be halted before disconnecting any portion of the SCSI bus. However, it is not necessary to bring the cluster down to do this if you are using a SCSI configuration that allows disconnection of a portion of the SCSI bus without losing termination.

SCSI bus configurations using SCSI in-line terminators or Y cables at each node, or using a SCSI device which auto-terminates its ports when disconnected (such as the MSA30 MI), can allow online repair.

1. Halt the node. You can use Serviceguard Manager to do this, or use the `cmhaltnode` command. Packages should fail over normally to other nodes.
2. Remove the SCSI cable from the card.
3. Remove the defective SCSI card.
4. Install the new SCSI 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. You must set the SCSI ID for the new card to be the same as the card it is replacing.
5. Attach the new SCSI card.
6. Add the node back into the cluster. You can use Serviceguard Manager to do this, or use the `cmrunnode` command.

Replacing LAN or Fibre Channel Cards

If a LAN or fibre channel card fails and the card has to be replaced, you can replace it online or offline depending on the type of hardware and operating system you are running. It is not necessary to bring the cluster down to do this.

Offline Replacement

Follow these steps to replace an I/O card off-line.

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 I/O card.
4. Install the new I/O 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.)

Online Replacement

If your system hardware supports hotswap I/O cards, you have the option of replacing the defective I/O card online, using the HP-UX `olrad` command. The new card must be exactly the same card type as the card you removed. Serviceguard will automatically recover a LAN card once it has been replaced and reconnected to the network.

For more information, see the `olrad(1m)` manpage and the *Interface Card OL* Support Guide* which as of the date of this manual can be found at http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01916176/c01916176.pdf. See also "Removing a LAN or VLAN Interface from a Node" (page 287).

NOTE: After replacing a Fibre Channel I/O card, it may necessary to reconfigure the SAN to use the World Wide Name (WWN) of the new Fibre Channel card if Fabric Zoning or other SAN security requiring WWN is used.

After Replacing the Card

After the online or offline 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 -c clustername config.ascii
   ```

2. Use the `cmapplyconf` command to apply the configuration and copy the new binary file to all cluster nodes:

   ```
   cmapplyconf -C config.ascii
   ```

This procedure updates the binary file with the new MAC address and thus avoids data inconsistency between the outputs of the `cmviewconf` and `lanscan` commands.

Replacing a Failed Quorum Server System

When a Quorum Server (QS) 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)` manpage 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.

   See the `qs(1)` manpage 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, these things can happen:

- These three commands will not work:

  ```
  cmquerycl -q
  cmapplyconf -C
  cmcheckconf -C
  ```

- 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.

NOTE: Make sure that the old Quorum Server system does not rejoin 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 `cmviewcl`
- Reviewing the LAN Configuration

NOTE: HP recommends you use Serviceguard Manager as a convenient way to observe the status of a cluster and the properties of cluster objects: from the System Management Homepage (SMH), select the cluster you need to troubleshoot.

Reviewing Package IP Addresses

The `netstat -in` command can be used to examine the LAN configuration. The command, if executed on `ftsys9` after `ftsys10` has been halted, shows that the package IP addresses are assigned to lan0 on `ftsys9` along with the primary LANIP address.

```
ftsys9>netstat -in
IPv4:
Name      Mtu  Network     Address         Ipkts   Ierrs Opkts   Oerrs Coll
ni0#      0    none        none            0       0     0       0     0
ni1*      0    none        none            0       0     0       0     0
lo0       4608 127         127.0.0.1       10114   0     10      0     0
lan0      1500 15.13.168   15.13.171.14    959269  0     33069   0     0
lan0:1    1500 15.13.168   15.13.171.23    959269  0     33069   0     0
lan0:2    1500 15.13.168   15.13.171.20    959269  0     33069   0     0
lan1*     1500 none        none            418623  0     55822   0     0
IPv6:
Name         Mtu Address/Prefix                                Ipkts    Opkts
lan1*        1500 none                                             0        0
lo0          4136 ::1/128                                      10690    10690
```

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`. Also, package-related messages are logged into the package log file. The package log file is located in the package

directory, by default. You can use a text editor, such as `vi`, or the `more` command to view the log file for historical information on your cluster.

It is always a good idea to review the `syslog.log` file on *each* of the nodes in the cluster when troubleshooting cluster problems.

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 *System Administrator's Guide* 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 management protocols.
Dec 14 14:33:48 star04 cmcld[2048]: Attempting to form a new cluster
Dec 14 14:33:53 star04 cmcld[2048]: 3 nodes have formed a new cluster
Dec 14 14:33:53 star04 cmcld[2048]: The new active cluster membership is:
    star04(id=1) , star05(id=2), star06(id=3)
Dec 14 17:33:53 star04 cmlvmd[2049]: Clvmd initialized successfully.
Dec 14 14:34:44 star04 CM-CMD[2054]: cmrunpkg -v pkg5
Dec 14 14:34:44 star04 cmcld[2048]: Request from node star04 to start
    package pkg5 on node star04.
Dec 14 14:34:44 star04 cmcld[2048]: Executing '/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 exited with
    NO_RESTART.
Dec 14 14:34:45 star04 cmcld[2048]: Examine the file
    /etc/cmcluster/pkg5/pkg5_run.log for more details.
```

The following is an example of a successful package starting:

```
Dec 14 14:39:27 star04 CM-CMD[2096]: cmruncl
Dec 14 14:39:27 star04 cmcld[2098]: Starting cluster management protocols.
Dec 14 14:39:27 star04 cmcld[2098]: Attempting to form a new cluster
Dec 14 14:39:27 star04 cmclconfd[2097]: Command execution message
Dec 14 14:39:33 star04 cmcld[2098]: 3 nodes have formed a new cluster
Dec 14 14:39:33 star04 cmcld[2098]: The new active cluster membership is:
   star04(id=1), star05(id=2), star06(id=3)
Dec 14 17:39:33 star04 cmlvmd[2099]: Clvmd initialized successfully.
Dec 14 14:39:34 star04 cmcld[2098]: Executing '/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 node 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.

Reviewing Serviceguard Manager Log Files

From the System Management Homepage (SMH), click `Tools`, then select `Serviceguard Manager`, select the cluster you are interested and then choose `View -> Operation Log`.

Reviewing the System Multi-node Package Files

If you are running Veritas Cluster Volume Manager and you have problems starting the cluster, check the log file for the system multi-node package. For CVM 4.1 and later, the file is `SG-CFS-pkg.log`.

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. By default, it is found at `/etc/cmcluster/<package_name>/control_script.log`; but another location may have been specified in the package configuration file's *`script_log_file`* parameter. This log documents all package run and halt activities. If you have written a separate run and halt script for a legacy 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 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
- Networking and security configuration errors
- 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

If you are having trouble starting Serviceguard, it is possible that someone has accidentally deleted, modified, or placed files in, the directory that is reserved for Serviceguard use only: $SGCONF/rc, by default /etc/cmcluster/rc on an HP-UX system.

Networking and Security Configuration Errors

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.

In many cases, a symptom such as `Permission denied...` or `Connection refused...` is the result of an error in the networking or security configuration. Most such problems can be resolved by correcting the entries in `/etc/hosts`. See "Configuring Name Resolution" (page 159) for more information.

Cluster Re-formations Caused by Temporary Conditions

You may see Serviceguard error messages, such as the following, which indicate that a node is having problems:

`Member node_name seems unhealthy, not receiving heartbeats from it.`

This may indicate a serious problem, such as a node failure, whose underlying cause is probably a too-aggressive setting for the *MEMBER_TIMEOUT* parameter; see the next section, "Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low". Or it may be a transitory problem, such as excessive network traffic or system load.

What to do: If you find that cluster nodes are failing because of temporary network or system-load problems (which in turn cause heartbeat messages to be delayed in network or during processing), you should solve the networking or load problem if you can. Failing that, you can increase the value of *MEMBER_TIMEOUT*, as described in the next section.

Cluster Re-formations Caused by MEMBER_TIMEOUT Being Set too Low

If you have set the *MEMBER_TIMEOUT* parameter too low, the cluster demon, `cmcld`, will write warnings to `syslog` that indicate the problem. There are three in particular that you should watch for:

1. `Warning: cmcld was unable to run for the last <n.n> seconds. Consult the Managing Serviceguard manual for guidance on setting MEMBER_TIMEOUT, and information on cmcld.`

   This means that `cmcld` was unable to get access to a CPU for a significant amount of time. If this occurred while the cluster was re-forming, one or more nodes could have failed. Some commands (such as `cmhaltnode (1m)`, `cmrunnode (1m)`, `cmapplyconf (1m)`) cause the cluster to re-form, so there's a chance that running one of these commands could precipitate a node failure; that chance is greater the longer the hang.

   What to do: If this message appears once a month or more often, increase *MEMBER_TIMEOUT* to more than 10 times the largest reported delay. For example, if the message that reports the

largest number says that `cmcld` was unable to run for the last 1.6 seconds, increase *`MEMBER_TIMEOUT`* to more than 16 seconds.

2. `This node is at risk of being evicted from the running cluster. Increase MEMBER_TIMEOUT.`

   This means that the hang was long enough for other nodes to have noticed the delay in receiving heartbeats and marked the node "unhealthy". This is the beginning of the process of evicting the node from the cluster; see "What Happens when a Node Times Out" (page 85) for an explanation of that process.

   What to do: In isolation, this could indicate a transitory problem, as described in the previous section. If you have diagnosed and fixed such a problem and are confident that it won't recur, you need take no further action; otherwise you should increase *`MEMBER_TIMEOUT`* as instructed in item 1.

3. `Member` *`node_name`* `seems unhealthy, not receiving heartbeats from it.`

   This is the message that indicates that the node has been found "unhealthy" as described in the previous bullet.

   What to do: See item 2.

For more information, including requirements and recommendations, see the *`MEMBER_TIMEOUT`* discussion under "Cluster Configuration Parameters " (page 105). See also "Modifying the MEMBER_TIMEOUT Parameter" (page 183) and "Cluster Daemon: cmcld" (page 39).

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 is 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/d*/*` - to check logical volumes 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 a failover package will not be transferred.
- The run or halt instructions may not run to completion.
- *auto_run* (automatic package 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.

△ **CAUTION:** Do not use the HP-UX `mount` and `umount` commands in a CFS cluster. Use `cfsmount` and `cfsumount` for legacy CFS packages; `cmhaltpkg` and `cmrunpkg` for modular CFS packages. Non-`cfs` commands (such as `mount -o cluster`, `dbed_chkptmount`, or `sfrac_chkptmount`) could cause conflicts with subsequent command operations on the file system or Serviceguard packages. These `mount` commands will not create an appropriate multi-node package, with the result that the cluster packages are not aware of the file system changes.

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; use the `cmmodnet(1m)` command.

   First determine which package IP addresses are installed by inspecting the output of `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` 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` 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 file systems. 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 (page 233) to `yes`. In this case, Serviceguard will cause the node where the control script timed out to halt (system reset). This effectively cleans up any side effects of the package's run or halt attempt (but remember that the system reset will cause *all* the packages running on that node to halt abruptly). In this case the package will be automatically restarted on any available alternate node for which it is configured. For more information, see "Responses to Package and Service Failures " (page 87).

Problems with Cluster File System (CFS)

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CFS (http://www.hp.com/go/hpux-serviceguard-docs).

If you have a system multi-node package for Veritas CFS, you may not be able to start the cluster until `SG-CFS-pkg` starts. Check `SG-CFS-pkg.log` for errors.

You will have trouble running the cluster if there is a discrepancy between the CFS cluster and the Serviceguard cluster. To check, use `gabconfig -a`.

The ports that must be up are:

1. `a` — `llt`, `gab`
2. `b` — `vxfen`
3. `v w` — `cvm`
4. `f` — `cfs`

You can also check the `syslog` file for information.

CAUTION: Do not use the HP-UX `mount` and `umount` commands in a CFS cluster. Use `cfsmount` and `cfsumount` for legacy CFS packages; `cmhaltpkg` and `cmrunpkg` for modular CFS packages. Commands such as `mount -o cluster`, `dbed_chkptmount`, or `sfrac_chkptmount` could cause conflicts with subsequent command operations on the file system or Serviceguard packages. Non-CFS `mount` commands will not create an appropriate multi-node package, with the result that the cluster packages are not aware of the file system changes.

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, logging an error such as the following 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 in the package log file

Follow the instructions in the messages 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, but they are caused specifically by errors in the control script for legacy packages. 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 a legacy package control script will cause additional details to be logged into the package log file, which can give you more information about 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 the Quorum Server

NOTE: See the *HP Serviceguard Quorum Server Version A.04.00 Release Notes* for information about configuring the Quorum Server. Do not proceed without reading the Release Notes for your version.

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 Quorum Server 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 problem; 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 *`MEMBER_TIMEOUT`* value. See "Cluster Configuration Parameters " (page 105) for more information about these parameters.

A message such as the following in a Serviceguard node's `syslog` file indicates that the node did not receive a reply to its lock request on time. This could be because of delay in communication between the node and the Quorum Server or between the Quorum Server and other nodes in the cluster:

```
Attempt to get lock /sg/cluser1 unsuccessful. Reason:
request_timedout
```

Messages

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 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.

The ECMT can be installed on HP-UX 11i v2, or 11i v3.

For more information, see the latest version of the *Enterprise Cluster Master Toolkit Release Notes* at the address given in the Preface (page 18).

A separate NFS toolkit is available. For more information, see the *Serviceguard NFS Toolkit Administrator's Guide* at the address given in the Preface (page 18).

Other application integration scripts are available from your HP representative.

B Designing Highly Available Cluster Applications

This appendix describes how to create or port applications for high availability, with emphasis on the following topics:

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 re-startable 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 re-startable 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 file systems, they must go through file systems recovery (`fsck`) before the data can be accessed. To help reduce this recovery time, the smaller these file systems are, the faster the recovery will be. Therefore, it is best to keep anything that can be replicated off the data file system. 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 file system. 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 file systems. Raw volumes do not require an `fsck` of the file system, 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 resource 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, use the procedure under "Using a Relocatable Address as the Source Address for an Application that is Bound to INADDR_ANY" (page 335).

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.

Using a Relocatable Address as the Source Address for an Application that is Bound to INADDR_ANY

△ **CAUTION:** The procedure in this section depends on setting the HP-UX kernel parameter *ip_strong_es_model*. HP supports setting this parameter for use with Serviceguard only if you are *not* using a cross-subnet configuration (page 29). Otherwise, leave the parameter at its default setting (zero, meaning disabled) and do not attempt to use the procedure that follows.

In an application for which INADDR_ANY is set, the procedure that follows enables HP-UX to use a relocatable package IP address on the same subnet as the source address of outgoing IP data packets, instead of automatically selecting the physical network IP address of the interface.

The procedure uses the HP-UX parameter *ip_strong_es_model* to enable per-interface default gateways. These default gateways are created for secondary network interfaces when you add a relocatable package IP address to the system. When the *ip_strong_es_model* is set to 1 and the sending socket (or communication endpoint) is bound to INADDR_ANY, IP will send the packet using the interface on which the inbound packet was received.

For more information about this parameter, see:

- The help menu for `ndd -h ip_strong_es_model`.
- The *HP-UX IPSec Version A.03.00 Administrator's Guide* which you can find at http://www.hp.com/go/hpux-security-docs —> HP-UX IPSec Software.

Perform the following steps on each node *before configuring the cluster:*

1. Enable strong end-system model permanently by editing /etc/rc.config.d/nddconf as follows:

   ```
   TRANSPORT_NAME[1]=ip
   NDD_NAME[1]=ip_strong_es_model
   NDD_VALUE[1]=1
   ```

2. If you have not already done so, disable dead gateway probing permanently by editing /etc/rc.config.d/nddconf as follows:

   ```
   TRANSPORT_NAME[2]=ip
   NDD_NAME[2]=ip_ire_gw_probe
   NDD_VALUE[2]=0
   ```

Once this has been done, use the HP-UX command `route (1m)` from within the package to add or delete a default route for each relocatable IP address, to allow it to communicate with all remote subnets. See the examples that follow.

⍟ **IMPORTANT:** You need to add and delete default routes only in a configuration in which the clients reside on a subnet different from that of the server's relocatable address. If all the client applications are on the same subnet as the relocatable IP address, you do not need to add or delete any routes for the relocatable addresses; they are added automatically when you add the relocatable addresses to the server.

For example, put a command such as the following in the customer_defined_run_commands function of a legacy package, or the start_command function in the *external_script* (page 239) for a modular package:

```
/usr/sbin/route add net default 128.17.17.1 1 source 128.17.17.17
```

In this example, 128.17.17.17 is the relocatable IP address of the package, and 128.17.17.1 is the gateway address of this network. So clients on any remote subnets coming into the 128.17.17.17 address will get 128.17.17.17 returned as the source IP address if the application in the package is bound to INADDR_ANY. This allows the IP packets to go through the firewall to reach other organizations on the network.

When the package halts, the route must be removed.

Put a command such as the following in the customer_defined_halt_commands function of a legacy package, or the stop_command function in the *external_script* (page 239) for a modular package:

```
/usr/sbin/route delete net default 128.17.17.1 1 source 128.17.17.17
```

Once the per-interface default route(s) have been added, `netstat -rn` would show something like the following, where `128.17.17.17` is the package relocatable address and `128.17.17.19` is the physical address on the same subnet:

```
Destination          Gateway            Flags Refs Interface  Pmtu
127.0.0.1             127.0.0.1          UH    0    lo0        32808
128.17.17.19          128.17.17.19       UH    0    lan5       32808
128.17.17.17          128.17.17.17       UH    0    lan5:1     32808
192.168.69.82         192.168.69.82      UH    0    lan2       32808
128.17.17.0           128.17.17.19       U     3    lan5       1500
128.17.17.0           128.17.17.17       U     3    lan5:1     1500
192.168.69.0          192.168.69.82      U     2    lan2       1500
127.0.0.0             127.0.0.1          U     0    lo0        32808
default               128.17.17.1        UG    0    lan5:1     1500
default               128.17.17.1        UG    0    lan5       1500
```

NOTE: If your package has more than one relocatable address on a physical interface, you must add a route statement for each relocatable address during package start up, and delete each of these routes during package halt.

For more information about configuring modular, packages, see Chapter 6 (page 216); for legacy packages, see "Configuring a Legacy Package" (page 289).

IMPORTANT: If you use a Quorum Server, make sure that you list all IP addresses or hostnames by which the nodes communicate with the Quorum Server in the authorization file `/etc/cmcluster/qs_authfile`.

For more information about the Quorum Server, see the latest version of the *HP Serviceguard Quorum Server Release Notes* at http://www.hp.com/go/hpux-serviceguard-docs —> `HP Serviceguard Quorum Server Software`.

Restoring Client Connections

How does a client reconnect to the server after a failure?

It is important to write client applications to specifically differentiate between the loss of a connection to the server and other application-oriented errors that might be returned. The application should take special action in case of connection loss.

One question to consider is how a client knows after a failure when to reconnect to the newly started server. The typical scenario is that the client must simply restart their session, or relog in. However, this method is not very automated. For example, a well-tuned hardware and application system may fail over in 5 minutes. But if users, after experiencing no response during the failure, give up after 2 minutes and go for coffee and don't come back for 28 minutes, the perceived downtime is actually 30 minutes, not 5. Factors to consider are the number of reconnection attempts to make, the frequency of reconnection attempts, and whether or not to notify the user of connection loss.

There are a number of strategies to use for client reconnection:

- Design clients which continue to try to reconnect to their failed server.

  Put the work into the client application rather than relying on the user to reconnect. If the server is back up and running in 5 minutes, and the client is continually retrying, then after 5 minutes, the client application will reestablish the link with the server and either restart or continue the transaction. No intervention from the user is required.

- Design clients to reconnect to a *different* server.

  If you have a server design which includes multiple active servers, the client could connect to the second server, and the user would only experience a brief delay.

  The problem with this design is knowing when the client should switch to the second server. How long does a client retry to the first server before giving up and going to the second server? There are no definitive answers for this. The answer depends on the design of the server

application. If the application can be restarted on the same node after a failure (see "Handling Application Failures " following), the retry to the current server should continue for the amount of time it takes to restart the server locally. This will keep the client from having to switch to the second server in the event of a application failure.

- Use a transaction processing monitor or message queueing software to increase robustness.

  Use transaction processing monitors such as Tuxedo or DCE/Encina, which provide an interface between the server and the client. Transaction processing monitors (TPMs) can be useful in creating a more highly available application. Transactions can be queued such that the client does not detect a server failure. Many TPMs provide for the optional automatic rerouting to alternate servers or for the automatic retry of a transaction. TPMs also provide for ensuring the reliable completion of transactions, although they are not the only mechanism for doing this. After the server is back online, the transaction monitor reconnects to the new server and continues routing it the transactions.

- Queue Up Requests

  As an alternative to using a TPM, queue up requests when the server is unavailable. Rather than notifying the user when a server is unavailable, the user request is queued up and transmitted later when the server becomes available again. Message queueing software ensures that messages of any kind, not necessarily just transactions, are delivered and acknowledged.

  Message queueing is useful only when the user does not need or expect response that the request has been completed (i.e, the application is not interactive).

Handling Application Failures

What happens if part or all of an application fails?

All of the preceding sections have assumed the failure in question was not a failure of the application, but of another component of the cluster. This section deals specifically with application problems. For instance, software bugs may cause an application to fail or system resource issues (such as low swap/memory space) may cause an application to die. The section deals with how to design your application to recover after these types of failures.

Create Applications to be Failure Tolerant

An application should be tolerant to failure of a single component. Many applications have multiple processes running on a single node. If one process fails, what happens to the other processes? Do they also fail? Can the failed process be restarted on the same node without affecting the remaining pieces of the application?

Ideally, if one process fails, the other processes can wait a period of time for that component to come back online. This is true whether the component is on the same system or a remote system. The failed component can be restarted automatically on the same system and rejoin the waiting processing and continue on. This type of failure can be detected and restarted within a few seconds, so the end user would never know a failure occurred.

Another alternative is for the failure of one component to still allow bringing down the other components cleanly. If a database SQL server fails, the database should still be able to be brought down cleanly so that no database recovery is necessary.

The worse case is for a failure of one component to cause the entire system to fail. If one component fails and all other components need to be restarted, the downtime will be high.

Be Able to Monitor Applications

All components in a system, including applications, should be able to be monitored for their health. A monitor might be as simple as a display command or as complicated as a SQL query. There must be a way to ensure that the application is behaving correctly. If the application fails and it is not detected automatically, it might take hours for a user to determine the cause of the downtime and recover from it.

Minimizing Planned Downtime

Planned downtime (as opposed to unplanned downtime) is scheduled; examples include backups, systems upgrades to new operating system revisions, or hardware replacements. For planned downtime, application designers should consider:

- Reducing the time needed for application upgrades/patches.

  Can an administrator install a new version of the application without scheduling downtime? Can different revisions of an application operate within a system? Can different revisions of a client and server operate within a system?

- Providing for online application reconfiguration.

  Can the configuration information used by the application be changed without bringing down the application?

- Documenting maintenance operations.

  Does an operator know how to handle maintenance operations?

When discussing highly available systems, unplanned failures are often the main point of discussion. However, if it takes 2 weeks to upgrade a system to a new revision of software, there are bound to be a large number of complaints.

The following sections discuss ways of handling the different types of planned downtime.

Reducing Time Needed for Application Upgrades and Patches

Once a year or so, a new revision of an application is released. How long does it take for the end-user to upgrade to this new revision? This answer is the amount of planned downtime a user must take to upgrade their application. The following guidelines reduce this time.

Provide for Rolling Upgrades

Provide for a "rolling upgrade" in a client/server environment. For a system with many components, the typical scenario is to bring down the entire system, upgrade every node to the new version of the software, and then restart the application on all the affected nodes. For large systems, this could result in a long downtime.

An alternative is to provide for a rolling upgrade. A rolling upgrade rolls out the new software in a phased approach by upgrading only one component at a time. For example, the database server is upgraded on Monday, causing a 15 minute downtime. Then on Tuesday, the application server on two of the nodes is upgraded, which leaves the application servers on the remaining nodes online and causes no downtime. On Wednesday, two more application servers are upgraded, and so on. With this approach, you avoid the problem where everything changes at once, plus you minimize long outages.

The trade-off is that the application software must operate with different revisions of the software. In the above example, the database server might be at revision 5.0 while the some of the application servers are at revision 4.0. The application must be designed to handle this type of situation.

For more information about the rolling upgrades, see "Software Upgrades " (page 343), and the Release Notes for your version of Serviceguard at http://www.hp.com/go/hpux-serviceguard-docs.

Do Not Change the Data Layout Between Releases

Migration of the data to a new format can be very time intensive. It also almost guarantees that rolling upgrade will not be possible. For example, if a database is running on the first node, ideally, the second node could be upgraded to the new revision of the database. When that upgrade is completed, a brief downtime could be scheduled to move the database server from the first node to the newly upgraded second node. The database server would then be restarted, while the first node is idle and ready to be upgraded itself. However, if the new database revision requires a different database layout, the *old* data will not be readable by the newly updated database. The downtime will be longer as the data is migrated to the new layout.

Providing Online Application Reconfiguration

Most applications have some sort of configuration information that is read when the application is started. If to make a change to the configuration, the application must be halted and a new configuration file read, downtime is incurred.

To avoid this downtime use configuration tools that interact with an application and make dynamic changes online. The ideal solution is to have a configuration tool which interacts with the application. Changes are made online with little or no interruption to the end-user. This tool must be able to do everything online, such as expanding the size of the data, adding new users to the system, adding new users to the application, etc. Every task that an administrator needs to do to the application system can be made available online.

Documenting Maintenance Operations

Standard procedures are important. An application designer should make every effort to make tasks common for both the highly available environment and the normal environment. If an administrator is accustomed to bringing down the entire system after a failure, he or she will continue to do so even if the application has been redesigned to handle a single failure. It is important that application documentation discuss alternatives with regards to high availability for typical maintenance operations.

C Integrating HA Applications with Serviceguard

The following is a summary of the steps you should follow to integrate an application into the Serviceguard environment:

1. Read the rest of this book, including the chapters on cluster and package configuration, and the Appendix "Designing Highly Available Cluster Applications."
2. Define the cluster's behavior for normal operations:
   - What should the cluster look like during normal operation?
   - What is the standard configuration most people will use? (Is there any data available about user requirements?)
   - Can you separate out functions such as database or application server onto separate machines, or does everything run on one machine?
3. Define the cluster's behavior for failover operations:
   - Does everything fail over together to the adoptive node?
   - Can separate applications fail over to the same node?
   - Is there already a high availability mechanism within the application other than the features provided by Serviceguard?
4. Identify problem areas
   - What does the application do today to handle a system reboot or panic?
   - Does the application use any system-specific information such as `uname()` or `gethostname()`, `SPU_ID`, or MAC address which would prevent it from failing over to another system?
5. Only applications specified by HP (or vendor approved) should be deployed in environments using an HP Serviceguard Storage Management Suite bundle with Veritas Cluster File System (CFS). If you are creating your own application for use in a CFS environment, some things to consider include:
   - Can the application be installed cluster-wide?
   - Does the application work with a cluster-wide file name space?
   - Will the application run correctly with the data (file system) available on all nodes in the cluster? This includes being available on cluster nodes where the application is not currently running.
   - Does the application have a robust shutdown procedure to guarantee it will be down on one node before it is started up on an other node?

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CFS (http://www.hp.com/go/hpux-serviceguard-docs).

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

Define a baseline behavior for the application on a standalone system:

1. 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/VxFS file systems as appropriate.
2. 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.
3. 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?
4. 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.
5. 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.
   a. Create the LVM infrastructure on the second system.
   b. Add the appropriate users to the system.
   c. Install the appropriate executables.
   d. 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?
   e. Repeat this process until you can get the application to run on the second system.
2. Configure the Serviceguard cluster:
   a. Create the cluster configuration.
   b. Create a package.
   c. Create the package script.
   d. 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.
4. If you will be building an application that depends on a Veritas Cluster File System (CFS) and Cluster Volume Manager (CVM), then consider the following:
   a. Build storage on all nodes of the cluster.
   b. Create the disk group and mount point packages.
   c. Make sure that your file systems are mounting and unmounting on the nodes as they are designed to do on your application layout.
   d. Once the `SG-CFS-DG-`*`ID#`* and `SG-CFS-MP-`*`ID#`* packages are running as desired, create your application packages, placing a dependency on the `SG-CFS-MP-`*`ID#`* package if desired. This is applicable in case of legacy CFS packages. For modular CFS packages, you can place the dependency of the application package on the consolidated modular package.

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CVM and CFS: http://www.hp.com/go/hpux-serviceguard-docs.

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 node2 back to node1.
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 file systems, 30 seconds to start the application and 3 minutes to recover the database.

D Software Upgrades

There are five types of upgrade you can do:

- rolling upgrade
- rolling upgrade using DRD
- non-rolling upgrade
- non-rolling upgrade using DRD
- migration with cold install

Each of these is discussed below.

Special Considerations for Upgrade to Serviceguard A.11.20

- Before you proceed, read the sections *Upgrading from an Earlier Serviceguard Release* and *Rolling Upgrade* in the latest version of the release notes for A.11.20
- Serviceguard A.11.20 is supported on HP–UX 11i v3 only.
- You can perform a rolling upgrade from A.11.19 to a later release, or from an earlier release to A.11.19, but you cannot do a rolling upgrade from a pre-A.11.19 release to a post-A.11.19 release.

  This is because A.11.19 is the only version of Serviceguard that will allow both the older version of the cluster manager and the new version (introduced in A.11.19) to coexist during a rolling upgrade.

  If you are upgrading from a pre-A.11.19 release: Start by reading *Upgrading from an Earlier Serviceguard Release* and *Rolling Upgrade* in the release notes. Then, if you decide to upgrade to A.11.19 in preparation for a rolling upgrade to A.11.20, continue with the subsection that follows; it provides information on, and instructions for, upgrading to A.11.19.

Special Considerations for Upgrade to Serviceguard A.11.19

You can skip this section if you are already running A.11.19.

Serviceguard A.11.19 introduces a new cluster manager. In a running cluster, this affects node timeout and failover and cluster re-formation; see the discussion of `MEMBER_TIMEOUT` under "Cluster Configuration Parameters " (page 105) for more information. There is also a one-time effect on upgrade to Serviceguard A.11.19.

Upgrade will trigger a cluster membership transition from the old to the new cluster manager when the last node in the cluster has been upgraded to A.11.19. This transition can take up to one second, during which time the old cluster manager will shut down and the new cluster manager will start.

△ **CAUTION:** From the time when the old cluster manager is shut down until the new cluster manager forms its first cluster, a node failure will cause the entire cluster to fail. HP strongly recommends that you use no Serviceguard commands other than `cmviewcl (1m)` until the new cluster manager successfully completes its first cluster re-formation. In the case of a Rolling Upgrade (page 344), this happens when the `cmrunnode` for the last node executes; in a Non-Rolling Upgrade (page 345), it happens during the initial `cmruncl`. See "How To Tell when the Cluster Re-formation Is Complete" for more information. For an upgrade to Serviceguard A.11.19, this recommendation is additional to the Limitations of Rolling Upgrades (page 346) and Limitations of Non-Rolling Upgrades (page 352), and supersedes any "Limitation" that is more lenient.

There may be further caveats that apply to specific upgrade paths; before you proceed, make sure you read the "Announcements", "Compatibility", and "Installing Serviceguard" sections of the latest version of the *HP Serviceguard Version A.11.19 Release Notes* at http://www.hp.com/go/hpux-serviceguard-docs .

If you are using a Quorum Server, make sure you also read the latest version of the *HP Serviceguard Quorum Server A.04.00 Release Notes* before you proceed. These are also posted on posted at http://www.hp.com/go/hpux-serviceguard-docs, under `HP Serviceguard Quorum Server Software`.

How To Tell when the Cluster Re-formation Is Complete

You can tell when the cluster has re-formed under the new cluster manager by monitoring `syslog` on any node. `syslog` will record that Serviceguard has started and finished upgrading the node to the new cluster manager (`Cluster Membership Protocol version 2`), and then, once all nodes have been upgraded, you will see a message indicating that the new cluster has formed. Watch for three messages similar to the following (for clarity, intervening messages have been replaced with ellipses `[...]`):

```
Nov 14 13:52:46 bbq1 cmcld[20319]: Starting to upgrade this node to Cluster Membership Protocol version 2
[....]
Nov 14 13:52:46 bbq1 cmcld[20319]: Finished upgrading this node to Cluster Membership Protocol version 2
[....]
Nov 14 13:52:47 bbq1 cmcld[20319]: Membership: membership at 3 is FORMED (coordinator 1) includes: 1 3 2 4
excludes
```

When you see the `Membership:` message, and specifically the capitalized word `FORMED` in this message, the transition to the new cluster manager is complete and you can use all the Serviceguard commands.

Types of Upgrade

Rolling Upgrade

In a rolling upgrade, you upgrade the HP-UX operating system (if necessary) and the Serviceguard software one node at a time without bringing down your cluster. A rolling upgrade can also be done any time one system needs to be taken offline for hardware maintenance or patch installations.

This method is among the least disruptive, but your cluster must meet both general and release-specific requirements. See "Guidelines for Rolling Upgrade" (page 345).

Rolling Upgrade Using DRD

DRD stands for Dynamic Root Disk. Using a Dynamic Root Disk allows you to perform the update on a clone of the root disk, then halt the node and reboot it from the updated clone root disk.

A rolling upgrade using DRD is like a rolling upgrade, but is even less disruptive because each node is down for a shorter time. It is also very safe; if something goes wrong you can roll back to the original (pre-upgrade) state by rebooting from the original disk.

This method is the least disruptive, but you need to make sure your cluster is eligible; see"Restrictions for DRD Upgrades" (page 345).

You can obtain the DRD software free from http://www.hp.com/go/drd.

Restrictions for DRD Upgrades

- Upgrade using DRD is supported only on HP-UX 11i v3.
- You must use the DRD software released with the September 2009 release of HP-UX 11i v3 or later.

  You can obtain the DRD software free from HP (see "Rolling Upgrade Using DRD"); you do not have to upgrade the operating system itself, so long as you are running 11i v3.
- The cluster must meet both general and release-specific requirements for a rolling upgrade; see "Guidelines for Rolling Upgrade" (page 345).
- Not all paths that are supported for rolling upgrade are supported for an upgrade using DRD, and there are additional requirements and restrictions for paths that are supported; see the Release Notes for the target version of Serviceguard.

Non-Rolling Upgrade

In a non-rolling upgrade, you halt the cluster before upgrading HP-UX (if necessary) and the Serviceguard software.

This method involves cluster down time, but allows for a broader range of upgrades than rolling upgrade. For example, you will need to do a non-rolling upgrade, or a migration with cold install, if you are upgrading from an older release of Serviceguard than rolling upgrade supports (see the Release Notes for the target version of Serviceguard for the specific rolling upgrade requirements).

Non-Rolling Upgrade Using DRD

In a non-rolling upgrade with DRD, you clone each node's root disk and apply the upgrade to the clone, then halt the cluster and reboot each node from its updated clone root disk.

This method involves much less cluster down time than a conventional non-rolling upgrade, and is particularly safe because the nodes can be quickly rolled back to their original (pre-upgrade) root disks. But you must make sure your cluster is eligible; see "Restrictions for DRD Upgrades" (page 345).

Migration with Cold Install

A cold install involves an installation or re-installation of HP-UX. It erases the current operating system and data and then installs the new operating system and software; you must then restore the data.

The advantage of migrating with cold install is that the software can be installed without regard for the software currently on the system or concern for cleaning up old software.

Guidelines for Rolling Upgrade

You can normally do a rolling upgrade if:

- You are not upgrading the nodes to a new version of HP-UX; *or*
- You *are* upgrading to a new version of HP-UX, but using the update process (`update-ux`), rather than a cold install.

  `update-ux` supports many, but not all, upgrade paths. For more information, see the *HP-UX Installation and Update Guide* for the target version of HP-UX.

IMPORTANT: Do not proceed without reading and understanding the following points:

- The above are general guidelines. The requirements for any particular Serviceguard release may be more restrictive. Rolling upgrade is supported only for the HP-UX/Serviceguard combinations listed in the Release Notes for the target version of Serviceguard. Make sure your cluster meets all specific requirements and limitations in those Release Notes, *and* all the general requirements and limitations in this Appendix, before you attempt a rolling upgrade.
- A rolling upgrade cannot include a cold install of HP-UX on any node. A cold install will remove configuration information; for example, device file names (DSFs) are not guaranteed to remain the same after a cold install.
- Until the 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.

Read this entire section, including the "Example of a Rolling Upgrade " (page 349), before you begin your upgrade.

You can perform a non-rolling upgrade (that is, an upgrade performed while the cluster is down) from any Serviceguard release to the latest Serviceguard release.

Performing a Rolling Upgrade

Limitations of Rolling Upgrades

The following limitations apply to rolling upgrades:

CAUTION: Stricter limitations apply to an upgrade to A.11.19; do not proceed with an upgrade to A.11.19 until you have read and understood the Special Considerations for Upgrade to Serviceguard A.11.19 (page 343).

- During a rolling upgrade to a release other than A.11.19, you can issue Serviceguard commands, but should execute 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 hardware configuration—including the cluster's network configuration—during rolling upgrade.
- You cannot modify the cluster or package configuration until the upgrade is complete.

  If you need to modify the configuration (for example, to take advantage of new features), upgrade all nodes to the new release, then modify the configuration file and copy it to all the nodes.

  NOTE: This means that you cannot migrate to the HP-UX 11i v3 agile addressing scheme for device files during a rolling upgrade if cluster lock disks are used as a tie-breaker, because that involves changing the cluster configuration. See "Updating the Cluster Lock Configuration" (page 281) for instructions in this case. See "About Device File Names (Device Special Files)" (page 77) for more information about agile addressing.
- 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.
- No more than two versions of Serviceguard can be running in the cluster while the rolling upgrade is in progress.
- Rolling upgrades are not intended as a means of using mixed releases of Serviceguard or HP-UX within the cluster. HP strongly recommends 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 a rolling upgrade is in progress.

Before You Start

Make sure you plan sufficient system capacity to allow moving the packages from node to node during the process without an unacceptable loss of performance.

△ **CAUTION:** Do not proceed with an upgrade to A.11.19 until you have read and understood the Special Considerations for Upgrade to Serviceguard A.11.19 (page 343).

Running the Rolling Upgrade

1. Halt the node you want to upgrade. You can do this in Serviceguard Manager, or use the `cmhaltnode` command. This will cause the node's packages to start up on an adoptive node
2. Edit `/etc/rc.config.d/cmcluster` to include the following line:

   ```
   AUTOSTART_CMCLD = 0
   ```

3. Upgrade the node to the new 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. See the section on hardware maintenance in the "Troubleshooting" chapter.

   For instructions on upgrading HP-UX, see the HP-UX Installation and Update Guide for the target version of HP-UX at http://www.hp.com/go/hpux-core-docs.

   To upgrade Serviceguard, use `swinstall (1m)` to install the new version. You do not need to remove the old version first, and you do not need to touch `/etc/cmcluster`.

4. Edit `/etc/rc.config.d/cmcluster` to include the following line:

   ```
   AUTOSTART_CMCLD = 1
   ```

   This will keep the upgraded node from automatically rejoining the cluster when it reboots.

5. If the Event Monitoring Service (EMS) is configured, restart it as follows:
   1. Kill all EMS monitors.
   2. Stop EMS clients.
   3. Kill all registrar processes.
   4. Kill the `p_client` demon.

   The `p_client` process restart immediately. The EMS registrar and monitor processes will be restarted automatically when they are needed.

   For more information, see "Using the Event Monitoring Service " (page 55).

6. Restart the cluster on the upgraded node. You can do this in Serviceguard Manager: from the System Management Homepage (SMH) choose `Tools -> Serviceguard Manager`, then select the node and choose `Administration -> Run Node...` Or, on the Serviceguard command line, issue the `cmrunnode` command.
7. Repeat this process for each node in the cluster.

If the cluster fails before the rolling upgrade is complete (because of a catastrophic power failure, for example), you can restart the cluster by entering the `cmruncl` command from a node which has been upgraded to the latest version of the software.

Keeping Kernels Consistent

If you change kernel parameters as a part of doing an upgrade, be sure to change the parameters to the same values on all nodes that can run the same packages in case of failover.

Migrating `cmclnodelist` entries from A.11.15 or earlier

Information in the `cmclnodelist` file is migrated to the new Access Control Policy form. All the `hostname 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.

"Controlling Access to the Cluster" (page 183) for more information about access control policies.

Performing a Rolling Upgrade Using DRD

IMPORTANT: All the limitations listed under "Guidelines for Rolling Upgrade" (page 345) and "Limitations of Rolling Upgrades " (page 346) also apply to a rolling upgrade with DRD. You should read the entire section on "Performing a Rolling Upgrade" (page 346) before you proceed.

Before You Start

CAUTION: Do not proceed with an upgrade to A.11.19 until you have read and understood the Special Considerations for Upgrade to Serviceguard A.11.19 (page 343).

IMPORTANT: Not all paths that are supported for rolling upgrade are supported for an upgrade using DRD, and there are additional requirements and restrictions for paths that are supported.

Do not proceed until you have read the "Announcements", "Compatibility", and "Installing Serviceguard" sections of the latest version of the Serviceguard Release Notes, made sure your cluster meets the current requirements, and taken any necessary preparation steps as instructed in the release notes.

- Make sure you plan sufficient system capacity to allow moving the packages from node to node during the process without an unacceptable loss of performance.
- Make sure you have read and understood the "Restrictions for DRD Upgrades" (page 345).
- Make sure that you have downloaded the latest DRD software and are thoroughly familiar with the DRD documentation. See "Rolling Upgrade Using DRD" (page 344) for more information.

Running the Rolling Upgrade Using DRD

1. On the node you want to upgrade, edit `/etc/rc.config.d/cmcluster` to include the following line:

   ```
   AUTOSTART_CMCLD = 0
   ```

   This value will be replicated on the clone root disk and will keep the upgraded node from automatically rejoining the cluster when it reboots.
2. Create a clone root disk for the node you want to upgrade.
3. Upgrade the clone disk to the new HP-UX release (if necessary), including Serviceguard (using `drd runcmd update-ux`), or upgrade Serviceguard alone (using `drd runcmd swinstall`).
4. Halt the node. You can do this in Serviceguard Manager, or use the `cmhaltnode (1m)` command. This will cause the node's packages to start up on an adoptive node.
5. Reboot the node from updated clone root disk.
6. Edit the `/etc/rc.config.d/cmcluster` file to include the following line:

   ```
   AUTOSTART_CMCLD = 1
   ```

7. Restart the cluster on the upgraded node, using Serviceguard Manager or `cmrunnode (1m)`.
8. Move the packages back to the upgraded node.
9. Verify that the applications are functioning properly.
   - If the applications do not function properly *and this is not the last node to be upgraded*, you can revert to the previous release on this node. This is called hot recovery, and it will allow you to keep the node up and running in the cluster while you upgrade the other nodes, though you will still need to investigate and solve the problems on this node before the cluster upgrade can complete. If you decide to perform a hot recovery, proceed as follows:
     1. Halt the packages and restart them on another node.
     2. Halt the node.
     3. Reboot the node from the original root disk (which still contains the pre-upgrade versions of HP-UX and Serviceguard).

4. Restart the cluster on the node, using Serviceguard Manager or `cmrunnode (1m)`.

- If the applications do not function properly and this is the last node to be upgraded, you cannot revert to the previous release on just this node. You must either solve the problems with this release on this node, or revert the entire cluster to the previous release by halting the cluster (`cmhaltcl`), rebooting *each* node from its original (pre-upgrade) root disk, and restarting the cluster (`cmruncl`).
- If the applications are functioning properly, continue with the next step.

10. Repeat the above steps for each node in the cluster.

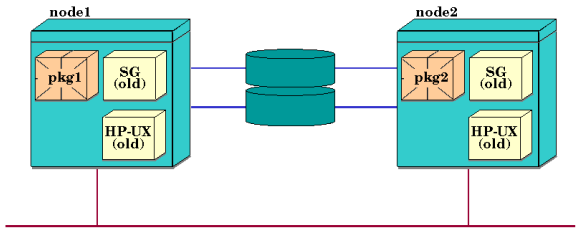
If the cluster fails before the rolling upgrade is complete (because of a catastrophic power failure, for example), you can restart the cluster by running `cmruncl (1m)` on a node which has been upgraded to the latest version of the software.

Example of a Rolling Upgrade

NOTE: Warning messages may appear during a rolling upgrade 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 41. (This and the following figures show the starting point of the upgrade as "SG (old)" and "HP-UX (old)", with a roll to "SG (new)" and "HP-UX (new)". Substitute the actual release numbers of your rolling upgrade path.)

Figure 41 Running Cluster Before Rolling Upgrade



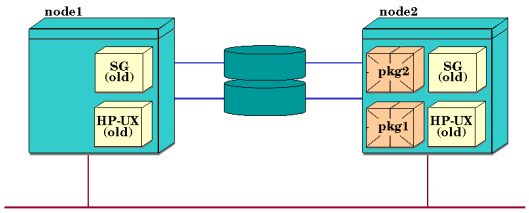
Step 1.

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 42.

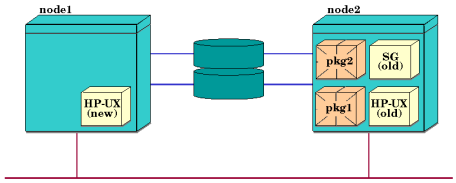
Figure 42 Running Cluster with Packages Moved to Node 2



Step 2.

Upgrade `node 1` to the next operating system release ("HP-UX (new)"), and install the next version of Serviceguard ("SG (new)").

Figure 43 Node 1 Upgraded to new HP-UX version



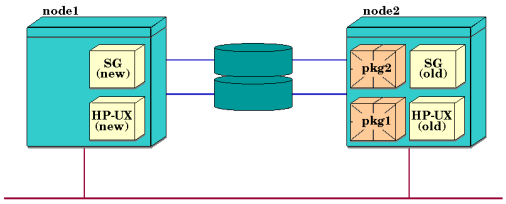
Step 3.

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 44.

Figure 44 Node 1 Rejoining the Cluster



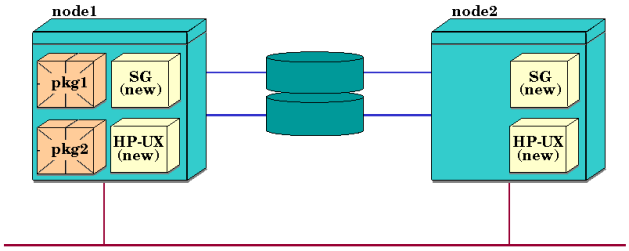
Step 4.

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 the new versions of HP-UX and Serviceguard.

Figure 45 Running Cluster with Packages Moved to Node 1



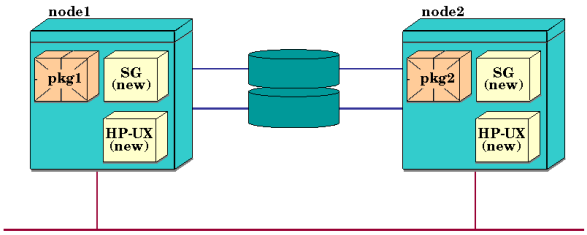
Step 5.

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 was disabled by the `cmhaltpkg` command. The final running cluster is shown in Figure 46.

Figure 46 Running Cluster After Upgrades



Guidelines for Non-Rolling Upgrade

Do a non-rolling upgrade if:

- Your cluster does not meet the requirements for rolling upgrade as specified in the Release Notes for the target version of Serviceguard; *or*
- The limitations imposed by rolling upgrades make it impractical for you to do a rolling upgrade (see "Limitations of Rolling Upgrades " (page 346)); *or*
- For some other reason you need or prefer to bring the cluster down before performing the upgrade.

Migrating Cluster Lock PV Device File Names

If you use cluster lock volumes, and you have decided to migrate the device file names used by the cluster nodes to the HP-UX 11i v3 agile addressing scheme or to cDSFs (see "About Device File Names (Device Special Files)" (page 77) and "About Cluster-wide Device Special Files (cDSFs)" (page 99)), use the procedures under "Updating the Cluster Lock Configuration" (page 281).

Other Considerations

See also "Keeping Kernels Consistent" and "Migrating `cmclnodelist` entries from A.11.15 or earlier" (page 347).

Performing a Non-Rolling Upgrade

Limitations of Non-Rolling Upgrades

△ **CAUTION:** Stricter limitations apply to an upgrade to A.11.19; do not proceed with an upgrade to A.11.19 until you have read and understood the Special Considerations for Upgrade to Serviceguard A.11.19 (page 343).

The following limitations apply to non-rolling upgrades:

- Binary configuration files may be incompatible between releases of Serviceguard. *Do not* manually copy configuration files between nodes.
- You must *halt the entire cluster* before performing a non-rolling upgrade.

Steps for Non-Rolling Upgrades

Use the following steps for a non-rolling software upgrade:

1. Halt all nodes in the cluster:

   ```
   cmhaltcl -f
   ```

2. If necessary, upgrade all the nodes in the cluster to the new HP-UX release. (See Step 3 under "Running the Rolling Upgrade" (page 347) for more information.)
3. Upgrade all the nodes in the cluster to the new Serviceguard release. (See Step 3 under "Running the Rolling Upgrade" (page 347) for more information.)
4. Restart the cluster:

   ```
   cmruncl
   ```

Performing a Non-Rolling Upgrade Using DRD

Limitations of Non-Rolling Upgrades using DRD

△ **CAUTION:** Stricter limitations apply to an upgrade to A.11.19; do not proceed with an upgrade to A.11.19 until you have read and understood the Special Considerations for Upgrade to Serviceguard A.11.19 (page 343).

IMPORTANT: Not all paths that are supported for upgrade are supported for an upgrade using DRD, and there are additional requirements and restrictions for paths that are supported.

Do not proceed until you have read the "Announcements", "Compatibility", and "Installing Serviceguard" sections of the latest version of the Serviceguard release notes, made sure your cluster meets the current requirements, and taken any necessary preparation steps as instructed in the release notes.

You must also make sure you have read and understood the "Restrictions for DRD Upgrades" (page 345).

In addition, all the limitations listed under "Limitations of Non-Rolling Upgrades" (page 352) apply to non-rolling upgrades using DRD. You should also read the "Guidelines for Non-Rolling Upgrade" (page 352) before you proceed.

Steps for a Non-Rolling Upgrade Using DRD

Use the following steps for a non-rolling software upgrade using DRD:

1. Create a clone root disk for each node.
2. Upgrade each clone disk to the new HP-UX release (if necessary), including Serviceguard (using `drd runcmd update-ux`), or upgrade Serviceguard alone (using `drd runcmd swinstall`).
3. Halt all nodes in the cluster:

   ```
   cmhaltcl -f
   ```

4. Reboot the nodes from their updated clone root disks.
5. Restart the cluster:

   ```
   cmruncl
   ```

6. Verify that the cluster is healthy, the packages can run, and applications are functioning properly.

If the cluster is not healthy or the packages and applications are not functioning properly, and you cannot fix the problems, you can revert to the original (pre-upgrade) cluster by halting the cluster (`cmhaltcl`), rebooting each node from its original (pre-upgrade) root disk, and restarting the cluster (`cmruncl`).

CAUTION: You must reboot all the nodes from their original disks before restarting the cluster; do not try to restart the cluster with some nodes booted from the upgraded disks and some booted from the pre-upgrade disks.

Guidelines for Migrating a Cluster with Cold Install

There may be circumstances when you prefer to do a cold install of the HP-UX operating system rather than an upgrade. A cold install erases the existing operating system and data and then installs the new operating system and software; you must then restore the data.

CAUTION: The cold install process erases the existing software, operating system, and data. If you want to retain any existing software, make sure you back up that software before migrating.

Checklist for Migration

Use the following as a checklist to prepare the migration.

CAUTION: This is a checklist, not a precise set of steps.

1. Back up the data, including databases, user and application data, volume group configurations, etc.

   NOTE: Data on shared disks, or on local disks in volumes that are not are touched by the HP-UX installation process, will not normally be erased by the cold install; you can re-import this data after the cold install. If you intend to do this, you must do the following before you do the cold install:

   - For LVM: create a map file for each LVM volume group and save it as part of your backup.
   - For VxVM: deport disk groups (halting the package should do this).

   See "Creating the Storage Infrastructure and file systems with LVM, VxVM and CVM" (page 168) for more information.
2. Halt the cluster applications, and then halt the cluster.
3. Do a cold install of the HP-UX operating system. For more information on the cold install process, see the HP-UX Installation and Update Guide for the target version of HP-UX at http://www.hp.com/go/hpux-core-docs.
4. Install any additional required software that did not come with your version of the HP-UX Operating Environment.
5. Install Serviceguard. The Serviceguard version must be compatible with the version of HP-UX installed in item 3 of this list. See the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix*, at http://www.hp.com/go/hpux-serviceguard-docs.
6. Recreate any user accounts needed for the cluster applications.
7. Recreate the cluster as described in Chapter 5: "Building an HA Cluster Configuration" (page 149).
8. Restart the cluster.
9. Reinstall the applications.
10. Restore or re-import the data.
11. Recreate and run the cluster packages as described in Chapter 6: "Configuring Packages and Their Services " (page 216).

E Blank Planning Worksheets

This appendix contains blank versions of the planning worksheets mentioned in"Planning and Documenting an HA Cluster " (page 88). 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   ___________

=============================================================================
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/*d*/*
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: _______________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: ____________________________________________________________

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: ____________________________________________________

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: _____ RAC Version: _____

    Node Names: _____

    Volume Groups (for packages):_____
    ============================================================================
    Subnets:
================================================================================
```

```
    Heartbeat Subnet: _____________________________________

    Monitored Non-heartbeat Subnet: _____________________

    Monitored Non-heartbeat Subnet: _____________________
=========================================----=====================================
    Cluster Lock: Volume Groups, LUN, or Quorum Server
=============================================================================
    Quorum Server:
    QS_HOST_____________________________QS_ADDR___________________
    QS_POLLING_INTERVAL_____________________________________
    QS_TIMEOUT_EXTENSION____________________________________
=============================================================================
    Volume Groups and Volumes:

    First Lock Volume Group:     |        Physical Volume/LUN:
                                 |
        _________________        |   Name on Node 1: _____________________
                                 |   Name on Node 2: _____________________
                                 |   Disk Unit No: __________
                                 |   Power Supply No: __________
=============================================================================
    Timing Parameters:
=============================================================================
    Member Timeout: ________
=============================================================================
    Network Polling Interval: _________
=============================================================================
Autostart Timeout: ____________
=============================================================================
   Access Policies:
    User name:
    Host node:
    Role:
=============================================================================
```

Package Configuration Worksheet

```
Package Configuration File Data:=============================================================Package Name:
_____________Package Type:_____________
Primary Node: ___________________
First Failover Node:_____________________
Additional Failover Nodes:_____________________________________
Run Script Timeout: _____ Halt Script Timeout: _____________
Package AutoRun Enabled?  ______ Local LAN Failover Allowed?  _____
Node Failfast Enabled? ________
Failover Policy:_____________________ Failback_policy:_____________________________________

Additional Package Resource:
Resource Name:___________ Polling Interval_______
Start____________________Resource UP Value__________

Access Policies:
User:_________________ From node:_______ Role:_____________
User:_________________ From node:_______ Role:_____________

Log level____ Log file:_____________________________________

Priority_____________ Successor_halt_timeout___________
dependency_name _____ dependency_condition ___
_____________________________________________________________________
LVM Volume Groups:vg_____________vg_____________vg_____________vg_____________
vgchange_cmd: ____________________________________________
CVM Disk Groups [ignore CVM items if CVM is not being used]:
cvm_vg___________cvm_dg____________cvm_vg_____________
cvm_activation_cmd: ____________________________________________
VxVM Disk Groups:
_________vxvm_dg____________vxvm_dg____________
vxvol_cmd  ____________________________________________

Logical Volumes and File Systems:
fs_name_____________________ fs_directory_____________fs_mount_opt___________
fs_umount_opt_____________fs_fsck_opt_____________fs_type_____________
fs_name_____________________fs_directory_____________fs_mount_opt___________
fs_umount_opt_____________ fs_fsck_opt_____________fs_type_____________
fs_name_____________________fs_directory_____________fs_mount_opt___________
fs_umount_opt_____________ fs_fsck_opt_____________fs_type_____________
fs_mount_retry_count: ___________fs_umount_retry_count:___________________
Concurrent vgchange operations: ____________________________________
Concurrent mount/umount operations: ____________________________________
```

```
Concurrent fsck operations: _____
Kill processes accessing raw devices?_____
================================================================================
Network Information:
IP ________ IP__________IP__________IP__________subnet ___________
IP__________IP__________IP__________IP__________subnet___________

IP_subnet_node_____________IP_subnet_node_____________IP_subnet_node__________
Monitored subnet:_____________________________monitored_subnet_access_____________
Monitored subnet:_____________________________monitored_subnet_access_____________
Cluster interconnect subnet [SGeRAC only]:_____________________________________
================================================================================
Service Name: _______ Command: _________ Restart:___ Fail Fast enabled:____
Service Name: _______ Command: _________ Restart: __ Fail Fast enabled:_____
Service Name: _______ Command: _________ Restart: __ Fail Fast enabled:_____
================================================================================
Package environment variable:_____________________________________________
Package environment variable:_____________________________________________
External pre-script:______________________________________________________
External script:__________________________________________________________
================================================================================
```

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CVM and CFS: http://www.hp.com/go/hpux-serviceguard-docs.

F 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) on systems that support it. Topics are as follows:

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 at http://www.hp.com/go/hpux-core-docs in the Veritas Volume Manager Administrator's Guide and the Veritas Volume Manager Migration Guide for your VxVM version. Refer to Chapter 5 of the present manual for information on creating 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.

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.

Information about VxVM installation are in the Veritas Installation Guide for your version of VxVM, available from http://www.hp.com/go/hpux-core-docs.

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. You should convert all the volume groups for a package at the same time.

It is assumed that VxVM software and an appropriate version of HP-UX and Serviceguard have been installed on the node, and that the node has rebooted and rejoined the cluster.

1. Halt the package that activates the volume group you wish 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 Migration Guide for your version, available from http://www.hp.com/go/hpux-core-docs. 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, if used, 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 a legacy package that will you use with the Veritas Volume Manager (VxVM) disk groups. If you decide to create a new package, see Chapter 6: "Configuring Packages and Their Services " (page 216).

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 let's say we have two volumes defined in each of the two disk groups from above, `lvol101` and `lvol102`, and `lvol201` and `lvol202`. These are mounted on `/mnt_dg0101` and `/mnt_dg0102`, and `/mnt_dg0201` and `/mnt_dg0202`, respectively.

     `/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/lvol102"
     LV[2]="/dev/vx/dsk/dg02/lvol201"
     LV[3]="/dev/vx/dsk/dg02/lvol202"

     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

NOTE: Check the *Serviceguard/SGeRAC/SMS/Serviceguard Manager Plug-in Compatibility and Feature Matrix* and the latest Release Notes for your version of Serviceguard for up-to-date information about support for CVM and CFS: http://www.hp.com/go/hpux-serviceguard-docs.

For instructions on configuring packages to use CVM disk groups for raw storage (that is, without CFS) see "Creating the Storage Infrastructure with Veritas Cluster Volume Manager (CVM)" (page 208).

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.

G Migrating from Legacy CFS Packages to Modular CFS Packages

To migrate from Legacy CFS packages to Modular CFS packages for use with the CVM or CFS on systems that support it, follow these steps:

1. Halt the application package:

   ```
   cmhaltpkg <app_package_name>
   ```

2. Unconfigure the CFS legacy disk group and the mount point package:

   ```
   cfsumount <mount point>
   cfsdgadm deactivate <disk group>
   cfsmntadm delete <mount point>
   cfsdgadm delete <disk group>
   ```

3. Configure the CFS modular disk group and mount point package. For details, see "Creating Modular Disk Group and Mount Point Packages" (page 196).
4. Start the application package.

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
- IPv6 Relocatable Address and Duplicate Address Detection Feature (page 367)

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 15 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 leftmost 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:

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:

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:

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:

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:

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:

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:

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. A value of "5" indicates that the scope is site-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 supports IPv6 for data and heartbeat IP.

To configure IPv6, the system should be set up in what is called a dual-stack configuration, which requires the IPv6 product bundle.

The restrictions for supporting IPv6 in Serviceguard are listed below.

- Auto-configured IPv6 addresses are not supported in Serviceguard as *HEARTBEAT_IP* or *STATIONARY_IP* addresses. 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, as package IPs, *HEARTBEAT_IP*s, or *STATIONARY_IP*s. 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_IP*s or *HEARTBEAT_IP*s can be mentioned in the cluster configuration file for a *NETWORK_INTERFACE*: one being the site-local IPv6 address, and the other being the global IPv6 address.

  NOTE: This restriction applies to *cluster* configuration, not *package* configuration: it does not affect the number of IPv6 relocatable addresses of the same scope type (site-local or global) that a package can use on an interface.

- 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.

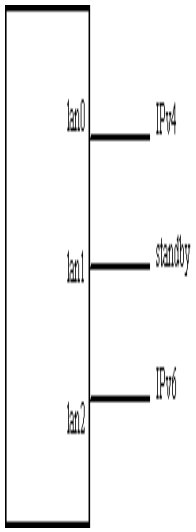
  That is, IPv4 and IPv6 addresses from two separate network interfaces are mutually exclusive in a failover condition.

- 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.

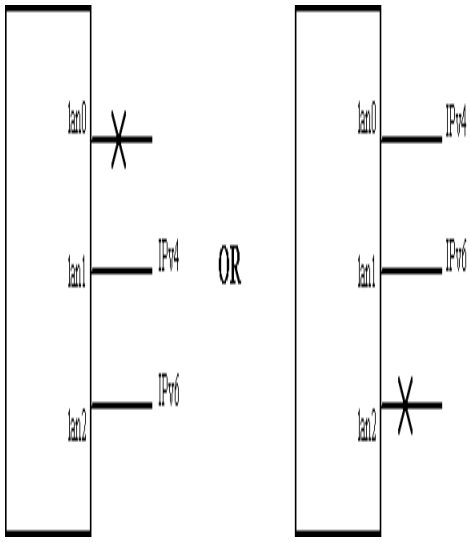
The examples that follow illustrate this.

Example Configurations

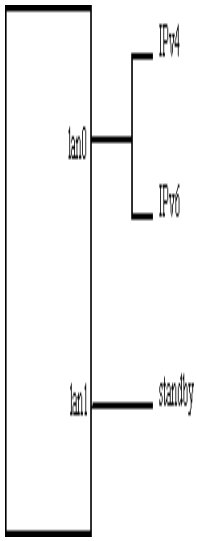
An example of a LAN configuration on a cluster node using both IPv4 and IPv6 addresses is shown in below.



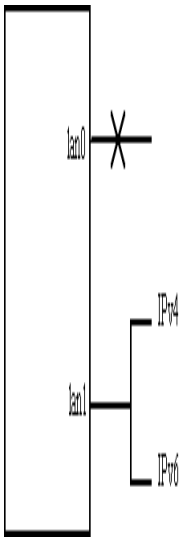
Following the loss of lan0 or lan2, lan1 can adopt either address, as shown below.



The same LAN card can be configured with both IPv4 and IPv6 addresses, as shown in below.



This type of configuration allows failover of both addresses to the standby. This is shown in below.


lan0
lan1
IPv4
IPv6

I Using Serviceguard Manager

HP Serviceguard Manager is a web-based, HP System Management Homepage (HP SMH) tool, that replaces the functionality of the earlier Serviceguard management tools. HP Serviceguard Manager allows you to monitor, administer and configure a Serviceguard cluster from any system with a supported web browser.

Serviceguard Manager does not require additional software installation. Instead, using your browser, you log into an HP Systems Management Homepage (SMH) and access the HP Serviceguard Manager tool, as well as other system management tools.

The HP Serviceguard Manager Main Page provides you with a summary of the health of the cluster including the status of each node and its packages.

About the Online Help System

Use this section to help you become familiar with Serviceguard Manager. Once Serviceguard Manager is running, use the tooltips by moving your mouse over a field from the read-only property pages for a brief definition for each field. You can also access the online help by clicking the  button located in the upper-right hand corner of the screen to view overview and procedure information. Start with the help topic Understanding the HP Serviceguard Manager Main Page. You should read the help topic About Security, as it explains HP Serviceguard Manager Access Control Policies, as well as `root` privileges.

Before Using HP Serviceguard Manager: Setting Up

NOTE: See the latest version of the Serviceguard Release Notes at http://www.hp.com/go/hpux-serviceguard-docs for the most up-to-date setup requirements.

You must have, or have done, the following before you can start using HP Serviceguard Manager:

- The hpuxswTOMCAT product.

  hpuxswTOMCAT is installed by default with HP-UX. Use the following command to check if it is on your system:

  ```
  swlist -l fileset | grep TOMCAT
  ```

- SMH (System Management Homepage; see the Release Notes for the required version)
- A web browser with access to SMH. See the Release Notes for supported browsers,
- Have launched SMH (`settings -> Security -> User Groups`) to configure user roles for SMH.
  - A user with HP SMH Administrator access has full cluster management capabilities.
  - A user with HP SMH Operator access can monitor the cluster and has restricted cluster management capabilities as defined by the user's Serviceguard role-based access configuration.
  - A user with HP SMH User access does not have any cluster management capabilities.

  See the online help topic `About Security` for more information.
- Have created the security "bootstrap" file `cmclnodelist`. See "Allowing Root Access to an Unconfigured Node" (page 157)for instructions.
- At least one cluster member node with Serviceguard and Serviceguard Manager installed.

Accessing Serviceguard Manager

The following table details the options you have to access Serviceguard Manager:

Table 16 Accessing Serviceguard Manager

Scenario	Scenario Use Case	Number of Clusters	Serviceguard Version	Serviceguard Manager Version
1	Single cluster management	1	A.11.20	B.03.10 B.03.00
			A.11.19	B.02.00
			A.11.18	B.01.01
			A.11.17.01	B.01.00
2	Multi-cluster management	more than 1 cluster	A.11.17.01 and later	HP SIM or multiple browser sessions of B.02.00 or later

Launching Serviceguard Manager

This section provides information about three common scenarios.

TIP: To reduce the time it takes to launch a session of HP Serviceguard Manager, do the following from the command line:

1. Stop `hpsmh`

   `/opt/hpsmh/bin/hpsmh stop`
2. Edit the file `/etc/rc.config.d/hpsmh` Set the value to the `START_TOMCAT` to `1. For example:`

   `START_TOMCAT=1`
3. Start `hpsmhd`.

   `/opt/hpsmh/bin/hpsmh autostart`

Scenario 1 - Single cluster management

Scenario 1 applies if the following is true:

- Manage a single cluster
- Have Serviceguard installed.

NOTE:

- A user with HP SMH Administrator access has full cluster management capabilities.
- A user with HP SMH Operator access can monitor the cluster and has restricted cluster management capabilities as defined by the user's Serviceguard role-based access configuration.
- A user with HP SMH User access does not have any cluster management capabilities.

1. Enter the standard URL "`http://<full hostname of server>:2301/`"

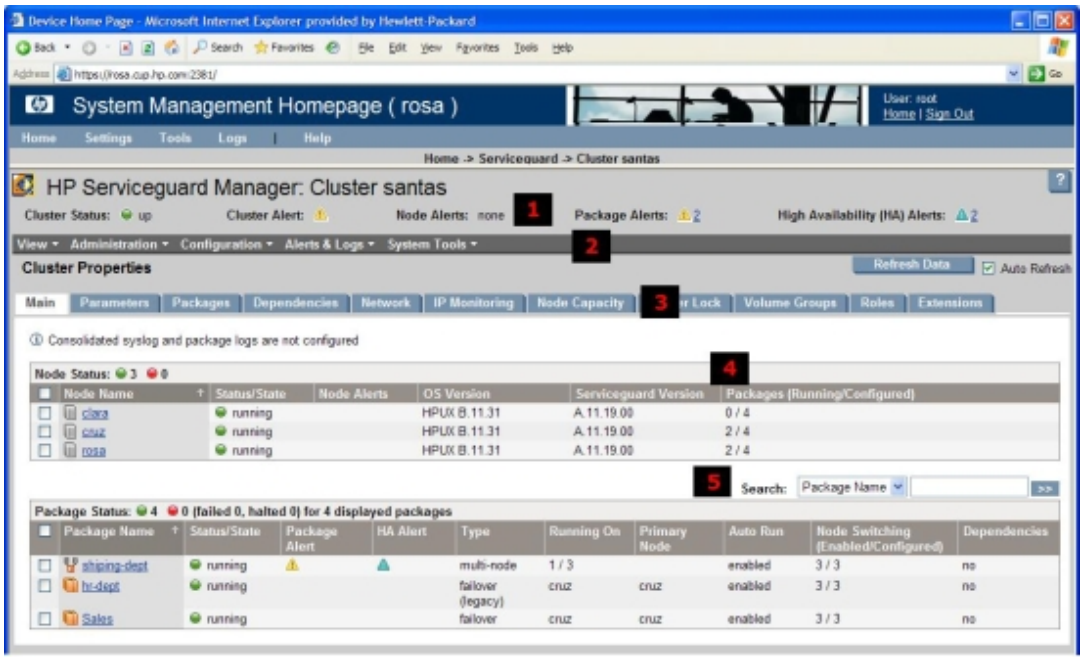
   For example `http://clusternode1.cup.hp.com:2301/`
2. When the System Management Homepage login screen appears, enter your login credentials and click Sign In.

   The System Management Homepage for the selected server appears.
3. From the Serviceguard Cluster box, click the name of the cluster.

NOTE: If a cluster is not yet configured, then you will not see the Serviceguard Cluster section on this screen. To create a cluster, from the SMH Tools menu, you must click Serviceguard Manager link in the Serviceguard box first, then click Create Cluster.

The figure below shows a browser session at the HP Serviceguard Manager Main Page.

Figure 47 System Management Homepage with Serviceguard Manager



Number	What is it?	Description
1	Cluster and Overall status and alerts	Displays information about the Cluster status, alerts and general information.
2	Menu tool bar	The menu tool bar is available from the HP Serviceguard Manager Homepage, and from any cluster, node or package view-only property page. Menu option availability depends on which type of property page (cluster, node or package) you are currently viewing.
3	Tab bar	The default Tab bar allows you to view additional cluster-related information. The Tab bar displays different content when you click on a specific node or package.
4	Node information	Displays information about the Node status, alerts and general information.
5	Package information	Displays information about the Package status, alerts and general information.

Scenario 2- Multi-Cluster Management for Serviceguard A.11.17 Cluster or Earlier

Open a separate browser session to administer, manage or monitor multiple clusters.

Scenario 2 applies if the following is true:

- One or more clusters running Serviceguard A.11.15.01 or later
- Serviceguard Manager version A.05.01 with HP SIM 5.10 or later installed on a server

NOTE: Serviceguard Manager can be launched by HP Systems Insight Manager version 5.10 or later if Serviceguard Manager is installed on an HP Systems Insight Manager Central Management Server.

For a Serviceguard A.11.17 cluster or earlier, Serviceguard Manager A.05.01 will be launched via Java Web Start. You must ensure that the hostname for each Serviceguard node can be resolved by DNS. For more information about this older version of Serviceguard Manager, see the Serviceguard Manager Version A.05.01 Release Notes at http://www.hp.com/go/hpux-serviceguard-docs.

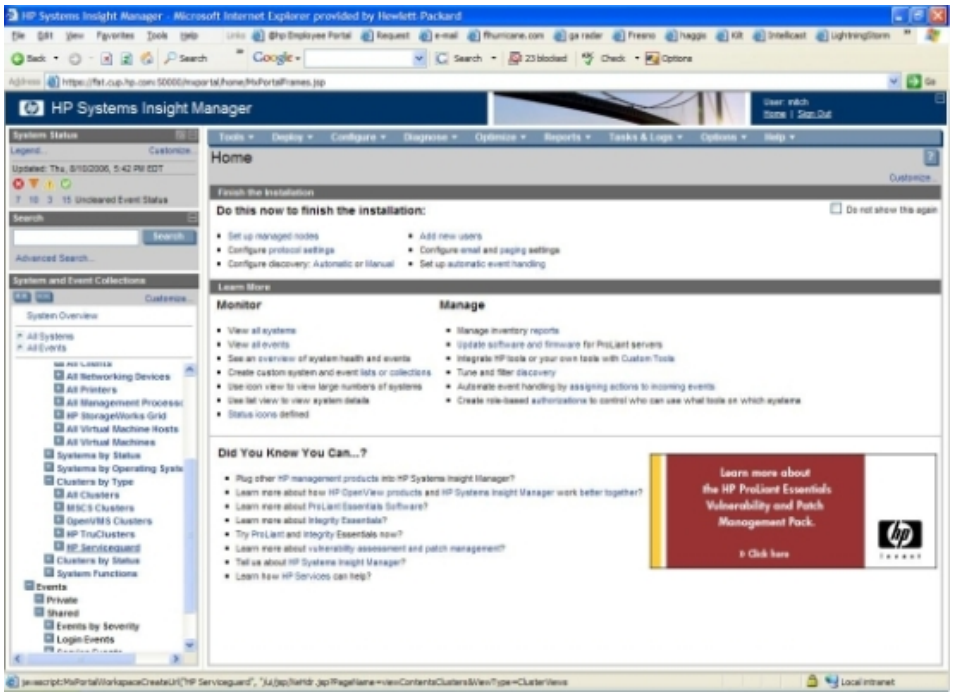
1. Enter the standard URL, `https://[full hostname of SIM server]:50000/`

For example `https://SIMserver.cup.hp.com:50000/`

2. When the Systems Insight Manager login screen appears, enter your login credentials and click **`Sign In`**.

3. From the left-hand panel, expand Cluster by Type.

Figure 48 Cluster by Type



4. Expand HP Serviceguard, and click on a Serviceguard cluster.

NOTE: If you click on a cluster running an earlier Serviceguard release, the page will display a link that will launch Serviceguard Manager A.05.01 (if installed) via Java Webstart.

Scenario 3- Multi-Cluster Management for Serviceguard A.11.17.01 Cluster or Later

This scenario is similar to Scenario 2 above. However, Scenario 3 applies if the following is true:

- One or more clusters running Serviceguard A.11.17.01 or later
- Serviceguard Manager version B.02.00 or later with HP SIM 5.10 or later installed on a server

NOTE: For a Serviceguard A.11.17.01 cluster or later, Systems Insight Manager will attempt to launch Serviceguard Manager from one of the nodes in the cluster.

J Maximum and Minimum Values for Parameters

Table 17 shows the range of possible values for cluster configuration parameters.

Table 17 Minimum and Maximum Values of Cluster Configuration Parameters

Cluster Parameter	Minimum Value	Maximum Value	Default Value	Notes
Member Timeout	See *MEMBER_TIMEOUT* under "Cluster Configuration Parameters" in Chapter 4.	See *MEMBER_TIMEOUT* under "Cluster Configuration Parameters" in Chapter 4.	14,000,000 microseconds	
AutoStart Timeout	60,000,000 microseconds	No Limit (ULONG_MAX)	600,000,000 microseconds	
Network Polling Interval	100,000 microseconds	No Limit (ULONG_MAX)	2,000,000 microseconds	
Maximum Configured Packages	0	300	300	

ULONG_MAX is a number equal to 4,294,967,295, which is therefore a practical limit.

Table 18 shows the range of possible values for package configuration parameters.

Table 18 Minimum and Maximum Values of Package Configuration Parameters

Package Parameter	Minimum Value	Maximum Value	Default Value	Notes
Run Script Timeout	10 seconds	4294 seconds if a non-zero value is specified	0 (NO_TIMEOUT)	This is a recommended value.
Halt Script Timeout	10 seconds	4294 seconds if a non-zero value is specified	0 (NO_TIMEOUT)	This is a recommended value, but note that the Halt Timeout value must be greater than the sum of all Service Timeout values
Service Halt Timeout	0 seconds	4294 seconds	0 (no time waited before the service is terminated)	

Index

D

E

F

G

H

T

U