Managing MC/ServiceGuard

B3936-90019

August 1997

Legal Notices

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

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

Corporate Offices:

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

Use, duplication or disclosure by the U.S. Government Department of Defense is subject to restrictions as set forth in paragraph (b)(3)(ii) of the Rights in Technical Data and Software clause in FAR 52.227-7013.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

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

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

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Printing History

Printing Date	Part Number	Edition
January 1995	B3936-90001	First
June 1995	B3936-90003	Second
December 1995	B3936-90005	Third
August 1997	B3936-90019	Fourth

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

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

HP Printing Division:

Enterprise Systems Division
Hewlett-Packard Co.
19111 Pruneridge Ave.
Cupertino, CA 95014

1 MC/ServiceGuard at a Glance

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

- What is MC/ServiceGuard?
- Using this Guide
- For More Information

If you are ready to start setting up MC/ServiceGuard clusters, skip ahead to the chapter "Planning and Documenting an HA Cluster." Specific steps for setup are given in the chapter "Building an HA Cluster Configuration."

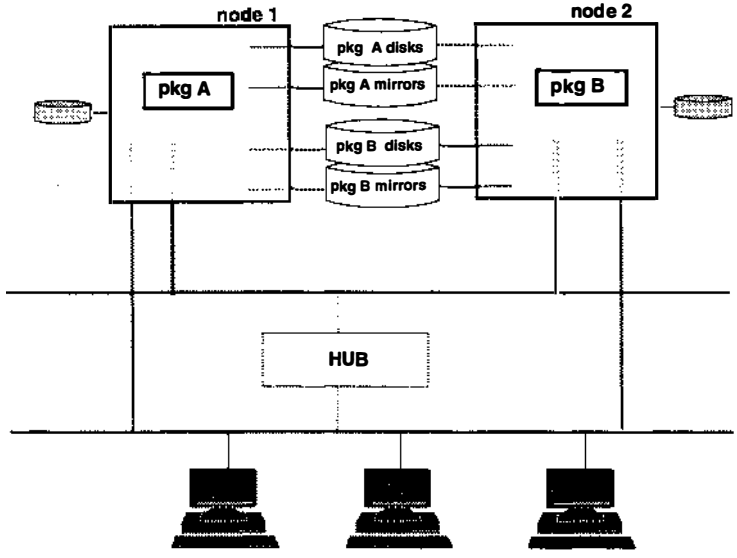
What is MC/ServiceGuard?

MC/ServiceGuard allows you to create high availability clusters of HP 9000 Series 800 computers. 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. MC/ServiceGuard and other high availability subsystems coordinate the transfer between components.

An MC/ServiceGuard **cluster** is a networked grouping of HP 9000 series 800 servers (host systems known as **nodes**) having sufficient redundancy of software and hardware that a **single point of failure** will not significantly disrupt service. Application services (individual HP-UX processes) are grouped together in **packages**; in the event of a single service, node, network, or other resource failure, MC/ServiceGuard can automatically transfer control of the package to another node within the cluster, allowing services to remain available with minimal interruption.

Figure 1-1 shows a typical MC/ServiceGuard cluster with two nodes.

Figure 1-1 **Typical Cluster Configuration**



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

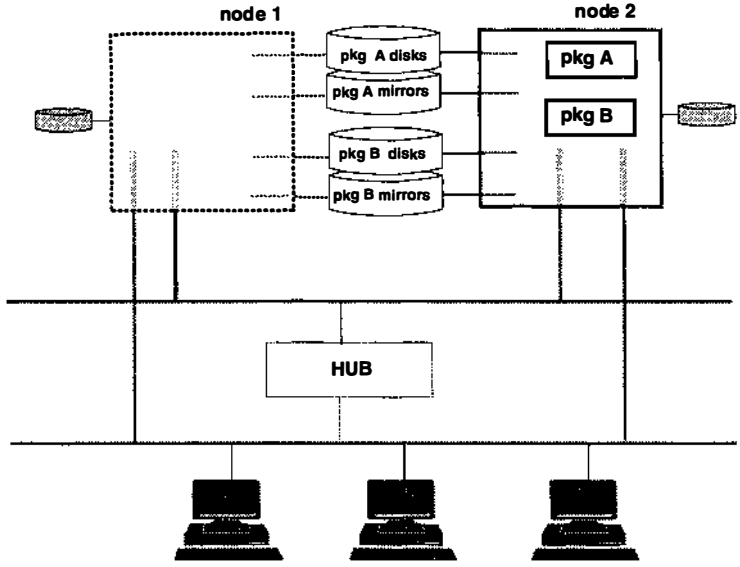
Mirror copies of data provide redundancy in case of disk failures. In addition, a total of four data buses are shown for the disks that are connected to node 1 and node 2. This configuration provides the maximum redundancy and also gives optimal I/O performance, since each package is using different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

Note that the network hardware is cabled to provide redundant LAN interfaces on each node. MC/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 MC/ServiceGuard Software.")

Failover

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

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



After this transfer, the package remains on the adoptive node as long the adoptive node continues running, even if the primary node comes back online. In situations where the adoptive node continues running successfully, you may manually transfer control of the package back to the primary node at the appropriate time. In certain circumstances, in the event of an adoptive node failure, a package that is running on an adoptive node will switch back automatically to its primary node (assuming the primary node is running again as a cluster member).

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

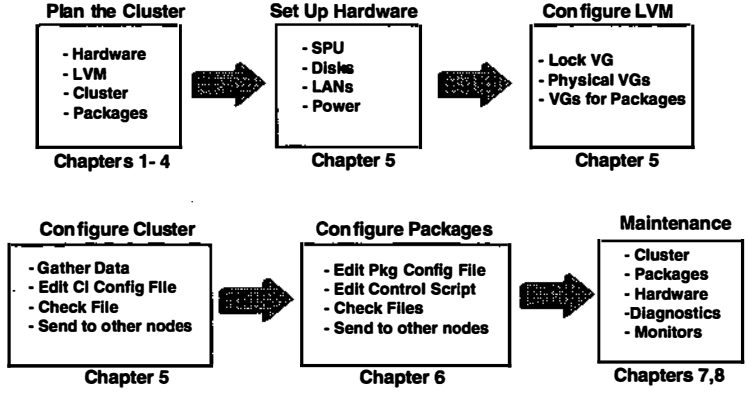
MC/ServiceGuard is designed to work in conjunction with other HP high availability products, such as MirrorDisk/UX, which provides disk redundancy to eliminate single points of failure in the disk subsystem; Event Monitoring Service (EMS), which lets you monitor and detect failures that are not directly handled by MC/ServiceGuard; disk arrays, which use various RAID levels for data protection;

and HP's UPS (uninterruptible power supply), PowerTrust, which eliminates failures related to power outage. These products are highly recommended along with MC/ServiceGuard to provide the greatest degree of availability.

Using this Guide

This manual presents the tasks you need to perform in order to create a functioning HA cluster using MC/ServiceGuard. These tasks are shown in Figure 1-3.

Figure 1-3 **Tasks in Configuring an MC/ServiceGuard Cluster**



The tasks in Figure 1-3 are covered in step-by-step detail in chapters 4 through 7. It is strongly recommended that you gather all the data that is needed for configuration *before you start*. Refer to Chapter 4, "Planning and Documenting an HA Cluster," for tips on gathering data.

If you are ready to set up an MC/ServiceGuard cluster, skip ahead to the "Planning" chapter. Specific guidelines on cluster creation are found in the chapters "Building a Cluster Configuration" and "Configuring Packages and their Services."

The following is a quick glance at the remaining chapters of this book:

- Chapter 2, "Understanding MC/ServiceGuard Hardware Configurations," describes the hardware configurations used by MC/ServiceGuard and provides a general view of how they work together.
- Chapter 3, "Understanding MC/ServiceGuard Software Components," describes the software components of MC/ServiceGuard and shows how they function within the HP-UX operating system.

- Chapter 4, "Planning and Documenting an HA Cluster," steps through the planning process and provides a set of worksheets for organizing information about the cluster.
- Chapter 5, "Building an HA Cluster Configuration," describes the creation of the cluster configuration.
- Chapter 6, "Configuring Packages and Their Services," describes the creation of high availability packages and the control scripts associated with them.
- Chapter 7, "Cluster and Package Maintenance," presents the basic cluster administration tasks.
- Chapter 8, "Troubleshooting Your Cluster," explains cluster testing and troubleshooting strategies.
- Appendix A, "MC/ServiceGuard Commands," lists the MC/ServiceGuard commands and reprints summary information from each man page.
- Appendix B, "Enterprise Cluster Master Toolkit," presents a set of package configuration files and control scripts for a group of Internet servers and third-party database products.
- Appendix C, "Designing Highly Available Cluster Applications," describes how to create or port applications for HA operation.
- Appendix D, "Integrating HA Applications with MC/ServiceGuard," summarizes the steps you follow to integrate an existing application into the MC/ServiceGuard environment.
- Appendix E, "Rolling Software Upgrades," describes the process of upgrading your MC/ServiceGuard software or the HP-UX operating system to a new release while the cluster is running.
- Appendix F, "Using OTS/9000 on the LAN," explains how to configure your MC/ServiceGuard cluster to use OTS/9000 on your LAN.
- Appendix G, "Blank Planning Worksheets," contains a set of empty worksheets for preparing an MC/ServiceGuard configuration.

For More Information

The following documents contain additional useful information:

- *Clusters for High Availability: a Primer of HP-UX Solutions.* HP Press, 1995 (ISBN 0-13-494758-4)
- *Configuring OPS Clusters with MC/LockManager* (B5158-90002)
- *Using EMS HA Monitors* (B5735-90001)

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

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

2 Understanding MC/ServiceGuard Hardware Configurations

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

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

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. MC/ServiceGuard provides this support as follows:

- In the case of LAN failure, MC/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.

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

The current maximum number of nodes supported in an MC/ServiceGuard cluster is 8. Fast/Wide SCSI disks or disk arrays can be connected to a maximum of 4 nodes at a time on a shared (multi-initiator) bus. Disk arrays using fibre channel and those that do not use a shared bus — such as the EMC Symmetrix — can be simultaneously connected to all 8 nodes.

The guidelines for package failover depend on the type of disk technology in the cluster. For example, a package that accesses data on a Fast/Wide SCSI disk or disk array can failover to a maximum of 4 nodes. A package that accesses data from a disk in a cluster using Fibre Channel or EMC Symmetrix disk technology can be configured to failover to 8 nodes.

Note that a package that does *not* access data from a disk on a shared bus can be configured to failover to however many nodes are 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

MC/ServiceGuard supports the following local area networks in high availability configurations:

- Ethernet
- Token Ring
- 100VG
- 10BaseT
- 100BaseT
- IEEE 802.3 (local switching not transparent)
- FDDI
- Fibre Channel (EPS nodes only)
- ATM (non-heartbeat only)

For Ethernet or IEEE 802.3 configurations, subnets must be either all Ethernet or all IEEE 802.3, respectively. NFS diskless clusters are not supported. A highly available NFS server can be configured as a package; see the document *Managing Highly Available NFS* (HP Part Number B5125-90001) for details.

Redundancy in Network Interfaces

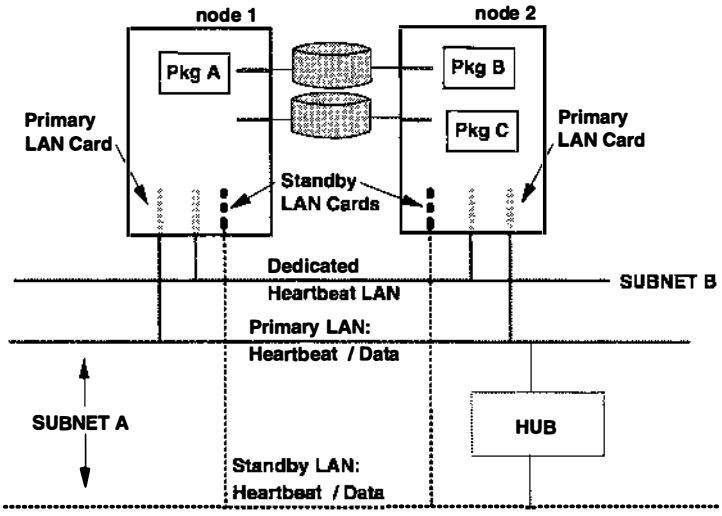
To eliminate single points of failure for networking, each subnet accessed by a cluster node is required to have redundant network interfaces. Redundant cables are also needed to protect against cable failures. Each interface card is connected to a different cable, and the cables themselves are connected by a connector such as a hub or a bridge. In the case of FDDI networks, each interface card is connected via a cable to a different concentrator. This arrangement of physical cables connected to each other via a bridge or concentrator or switch is known as a **bridged net**.

IP addresses can be associated with interfaces on a bridged net. An interface that has an IP address associated with it is known as a **primary interface**, and an interface that does not have an IP address associated with it is known as a **standby interface**. Standby interfaces are those which are available for switching by MC/ServiceGuard if a failure occurs on the primary. When MC/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.

Redundant Ethernet Configuration

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

Figure 2-1 **Redundant LANs**



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

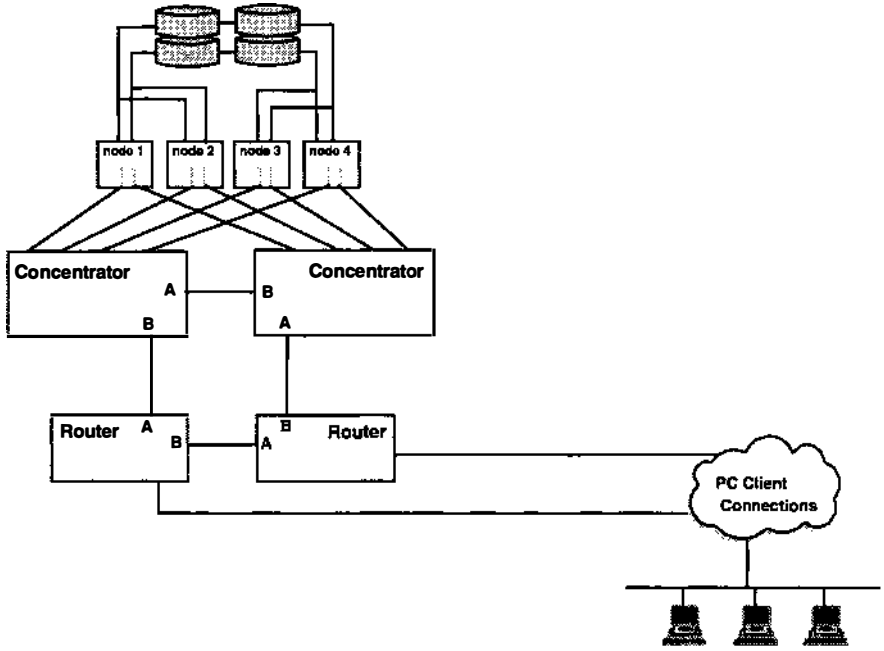
Redundant heartbeat is provided by the primary LAN and the dedicated LAN which are both carrying the heartbeat. In Figure 2-1, local switching is not needed for the dedicated heartbeat LAN, since there is already a redundant path via the other subnet. In case of data congestion on the primary LAN, the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

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

Providing Redundant FDDI Connections

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

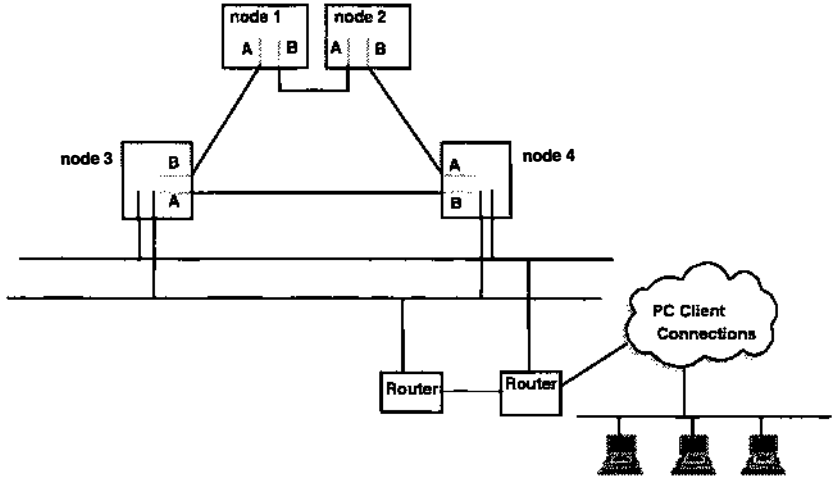
Figure 2-2 Redundant FDDI Configuration



Using Dual Attach FDDI Stations

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

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

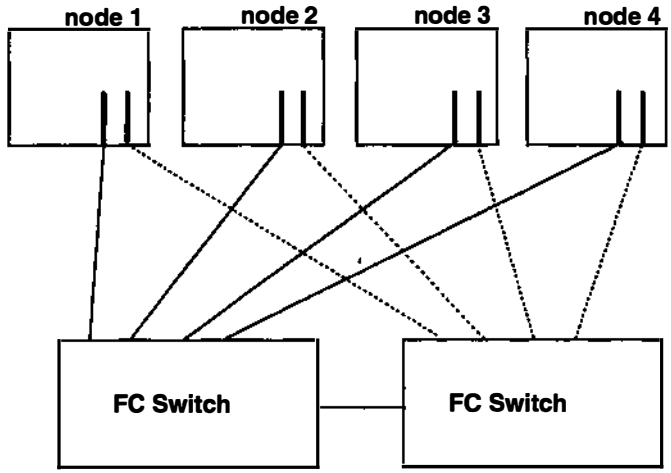


The use of dual attach cards gives protection against failures in both cables and connectors, but does not protect against card failures. LAN card failure would result in a package switching to another node.

Fibre Channel Switched Configurations

Another type of high speed fiber optic interconnect is provided with Fibre Channel, which is normally used with larger clusters (4 to 8 nodes). MC/ServiceGuard clusters configured on enterprise parallel server (EPS) systems can use Fibre Channel switches to provide redundant network paths among nodes. A four node example with two switches is shown in Figure 2-4. For a two-switch topology, all primary paths must go into one switch while all standbys must go to the second switch.

Figure 2-4 **FibreChannel Interconnect in a Four-Node Cluster**



In the figure, the solid lines refer to the primary network path among the nodes and the dotted lines refer to the standby path. If there is a failure in one of the primary lines, a local switch takes place on the node that has a failed primary line, and the standby is used instead.

Using a Serial (RS232) Heartbeat Line

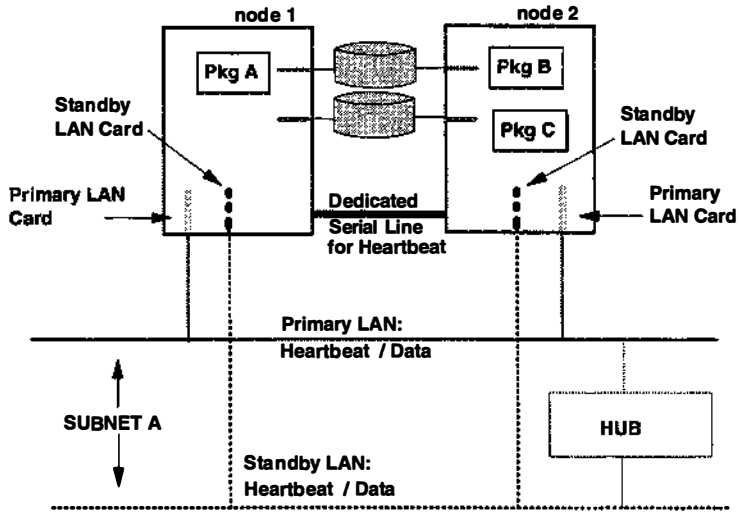
MC/ServiceGuard supports a two-node configuration using serial (RS232) communication for heartbeats only. You select this as an alternate heartbeat interface to provide redundant heartbeat. If you configure serial (RS232) as a heartbeat line, MC/ServiceGuard will send the heartbeat continuously on both the LAN configured for heartbeat and a monitored serial (RS232) line.

Even if you have a serial (RS232) line configured for redundant heartbeat, one LAN is still required to carry a heartbeat signal. The serial line heartbeat protects against network saturation but not against network failure, since MC/ServiceGuard requires TCP/IP to communicate between cluster members. If both network cards fail on one node, then having a serial line heartbeat keeps the cluster up long enough to detect the LAN controller card status and to fail the node with bad network connections while the healthy node stays up and runs all the packages.

NOTE The use of a serial (RS232) heartbeat line is supported only in a two-node cluster configuration.

A serial (RS232) heartbeat line is shown in Figure 2-5.

Figure 2-5 **Serial (RS232) Heartbeat Line**



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 the Logical Volume Manager. A disk volume group can be activated by no more than one node at a time, but when the package is moved, the volume group can be activated by the adoptive node. All of the disks in the volume group owned by a package must be connected to the original node and to all possible adoptive nodes for that package. Disk storage is made redundant by using RAID or software mirroring.

Supported Disk Interfaces

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

- Single-ended SCSI.
- Fast/Wide SCSI.
- HP FiberLink.
- FibreChannel.

Not all SCSI disks are supported. See the *HP 9000 Servers Configuration Guide* (available through your HP representative) for a list of currently supported disks.

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

When planning and assigning SCSI bus priority, remember that one node can dominate a bus shared by multiple nodes, depending on what SCSI addresses are assigned to the controller for each node on the shared bus. All SCSI addresses, including the addresses of all interface cards, must be unique for all devices on a shared bus. See the manual *Configuring HP-UX for Peripherals* for information on SCSI bus addressing and priority.

Data Protection

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

- Disk Mirroring
- Disk Arrays using RAID Levels and PV Links

Disk Mirroring

Disk mirroring is one method for providing data protection. The logical volumes used for MC/ServiceGuard packages should be mirrored. MC/ServiceGuard does not provide protection for data on your disks; this is provided by HP's MirrorDisk/UX product, which operates in conjunction with Logical Volume Manager. When you configure logical volumes using MirrorDisk/UX, the members of each mirrored set contain exactly the same data. If one disk should fail, MirrorDisk/UX will automatically keep the data available by accessing the other mirror. Three-way mirroring may be used to allow for online backups or even to provide an additional level of high availability.

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

While it is valuable to mirror your root disks, it is critical for high availability that you mirror your data disks. In the event of damage to a root disk of one SPU, the other SPUs in the cluster will take over control of the applications. However, if a data disk is damaged, and not mirrored, any application which depends on that disk will not be available until the problem with the disk is repaired. Even then, it may not be possible to recover the data on that disk.

Disk Arrays using RAID Levels and PV Links

An alternate method of achieving protection for your data is to use a disk array using 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 SCSI interfaces between each node and the array. The use of redundant interfaces, configured with LVM's PV Links feature protects against single points of failure in the I/O channel, and RAID 1 or 5 configuration provides redundancy for the storage media. (PV links are also known as alternate links in LVM.)

Monitoring of Disks Through Event Monitoring Service

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 (B5735AA). Monitoring can be set up to trigger a package failover or to report disk failure events to a target application such as ClusterView. Refer to the manual *Using EMS HA Monitors* (HP part number B5735-90001) for additional information.

Replacement of Failed Disk Mechanisms

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

Sample Disk Configurations

Figure 2-6 shows a two node cluster. Each node has one root disk which is mirrored and one package for which it is the primary node. Resources have been allocated to each node so that each node may adopt the package from the other node. Each package has one disk volume group assigned to it and the logical volumes in that volume group are mirrored. Please note that Package A's disk and the mirror of Package B's disk are on one interface while Package B's disk and the mirror of Package A's disk are on a separate bus. This arrangement eliminates single points of failure and makes either the disk or its mirror available in the event one of the buses fails.

Figure 2-6 **Mirrored Disks Connected for High Availability**

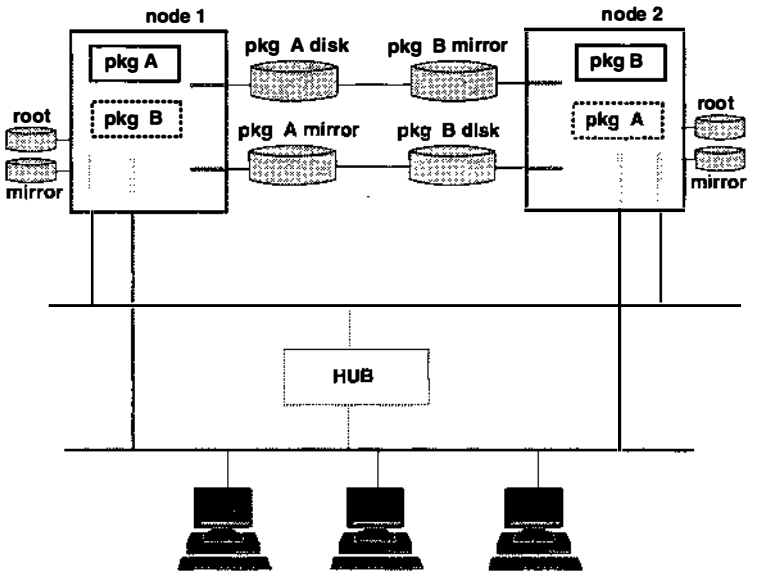
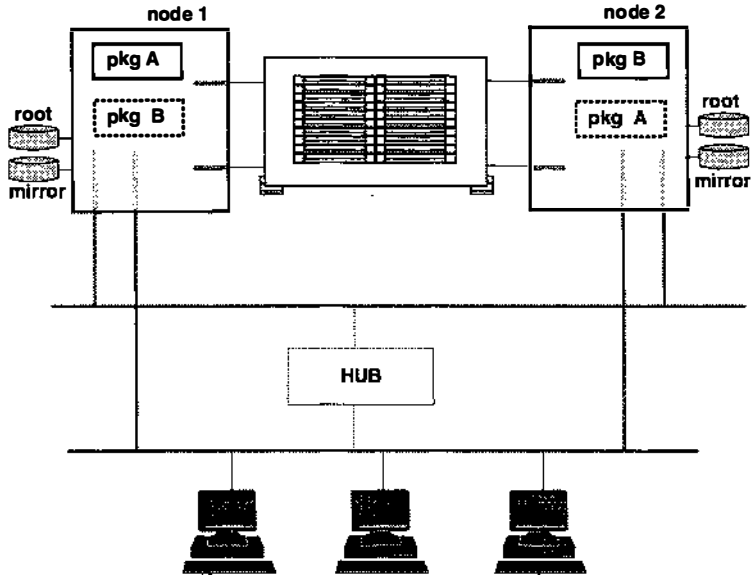


Figure 2-7 shows a similar cluster with a disk array connected to each node on two I/O channels. In this configuration, Logical Volume Manager's PV links are used to define the separate pathways to the data from one node.

Figure 2-7 **Cluster with High Availability Disk Array**



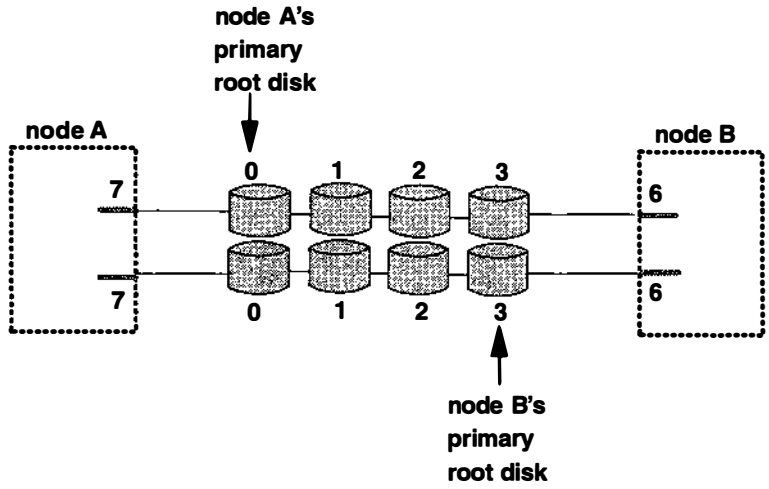
Details on logical volume configuration for MC/ServiceGuard, including PV Links, are given in the chapter "Building an HA Cluster Configuration."

Root Disk Limitations on Shared Buses

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

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

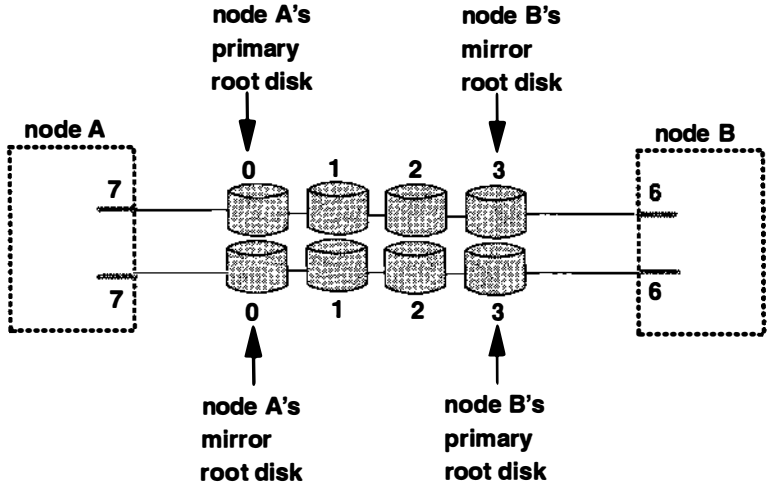
Figure 2-8 **Root Disks on Different Shared Buses**



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

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

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



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

Larger Clusters

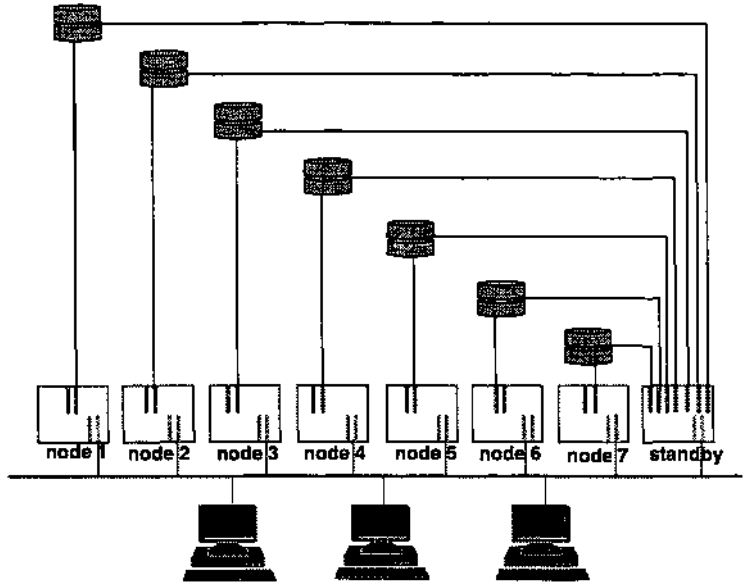
You can create clusters of up to 8 nodes with MC/ServiceGuard. Clusters of up to 8 nodes may be built by connecting individual SPUs via Ethernet; and you can configure up to 8 Enterprise Parallel Server (EPS) systems as an MC/ServiceGuard cluster using FDDI or Fibre Channel. EPS solutions are factory-packaged groups of rack-mounted systems that use a high-speed interconnect for communication among the SPU's. When configured as a high availability cluster, the EPS can use the FDDI or Fibre Channel link for heartbeats as well.

The possibility of configuring a cluster consisting of 8 nodes does not mean that all 8 nodes can be connected to the same disks using the same I/O bus. In the case of F/W SCSI, the practical limit on the number of nodes that can be attached to the same bus is four, because of bus loading and limits on cable length. However, 8 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.

Active/Standby Model

An eight node configuration in which one node acts as the standby for the other seven can easily be set up by equipping the backup node with seven shared buses allowing separate connections to each of the active nodes. This configuration is shown in Figure 2-10.

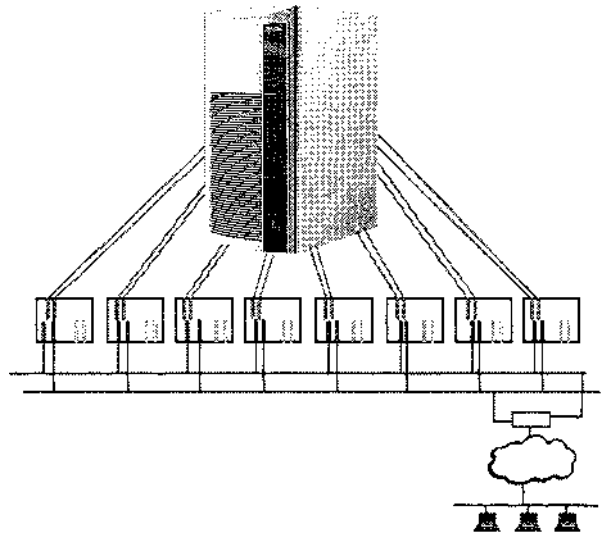
Figure 2-10 **Eight-Node Active/Standby Cluster**



Point to Point Connections to Storage Devices

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

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



For additional information about supported cluster configurations, refer to the *HP 9000 Servers Configuration Guide*, available through your HP representative.

3 Understanding MC/ServiceGuard Software Components

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

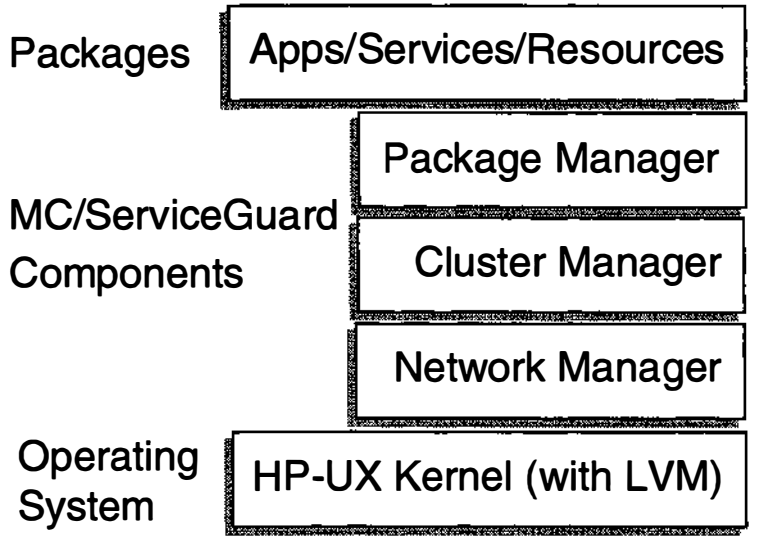
- MC/ServiceGuard Architecture
- How the Cluster Manager Works
- How the Package Manager Works
- How the Network Manager Works
- Responses to Failures

If you are ready to start setting up MC/ServiceGuard clusters, skip ahead to the "Planning" chapter.

MC/ServiceGuard Architecture

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

Figure 3-1 **MC/ServiceGuard Software Components**



How the Cluster Manager Works

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

Configuration of the Cluster

The system administrator sets up cluster configuration parameters and does an initial cluster startup; thereafter, the cluster regulates itself without manual intervention in normal operation. Configuration parameters for the cluster include the cluster name and nodes, networking parameters for the cluster heartbeat, cluster lock disk information, and timing parameters (discussed in detail in the "Planning" chapter). Cluster parameters are entered using SAM or by editing an ASCII cluster configuration template file. 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.

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 SAM 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.

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 sends a heartbeat message over over a stationary IP address on a monitored LAN or a serial (RS232) line to the cluster coordinator. (LAN monitoring is further discussed later in the section "Monitoring LAN Interfaces and Detecting Failure.")

The cluster coordinator looks for this message from each node, and if it is not received within the prescribed time, will re-form the cluster. At the end of the re-formation, if a new set of nodes form a cluster, that information is passed to the **package coordinator** (described further below, under "How the Package Manager Works"). Packages which were running on nodes that are no longer in the new cluster are transferred to their adoptive nodes in the new configuration. Note that if there is a transitory loss of heartbeat, the cluster may re-form with the same nodes as before. In such cases, packages do not halt or switch, though the application may experience a slight performance impact during the re-formation.

If heartbeat and data are sent over the same LAN subnet, data congestion may cause MC/ServiceGuard to miss heartbeats during the period of the heartbeat timeout and initiate a cluster re-formation that would not be needed if the congestion had not occurred. To prevent this situation, it is recommended that you have a dedicated heartbeat as well as configuring heartbeat over the data network or running heartbeat over a serial (RS232) line. A dedicated LAN is not required, but you may wish to use one if analysis of your networks shows a potential for loss of heartbeats in the cluster.

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

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

Automatic Cluster Restart

An automatic cluster restart occurs when all nodes in a cluster have failed. This is usually the situation when there has been an extended power failure and all SPUs went down. In order for an automatic cluster restart to take place, all nodes specified in the cluster configuration file must be up and running, must be trying to form a

cluster, and must be able to communicate with one another. Automatic cluster restart will take place if the flag `AUTOSTART_CMCLD` is set to 1 in the `/etc/rc.config.d/cmcluster` file.

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:

- 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 for Re-formation

The algorithm for cluster re-formation generally requires a cluster quorum of a strict majority (that is, more than 50%) of the nodes previously running. However, exactly 50% of the previously running nodes may re-form as a new cluster provided there is a guarantee that the other 50% of the previously running nodes do not also re-form. In these cases, a tie-breaker is needed. For example, if there is a communication failure between the nodes in a two-node cluster, and each node is attempting to re-form the cluster, then MC/ServiceGuard only allows one node to form the new cluster. This is ensured by using a **cluster lock**.

Use of the Cluster Lock

The cluster lock is a disk area located in a volume group that is shared by all nodes in the cluster. The cluster lock volume group and physical volume names are identified in the cluster configuration file. The cluster lock is used as a tie-breaker only for situations in which a running cluster fails and, as MC/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 the cluster lock. If communications are lost between these two nodes, the node with the cluster lock will take over the cluster and the other node will shut down. 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.

You can choose between two cluster lock options — a single or dual cluster lock — based on the kind of high availability configuration you are building. *A single cluster lock is recommended where possible.* With both single and dual locks, however, it is important that the cluster lock disk be available even if one node loses power; 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.If you have a cluster with more than 4 nodes, a cluster lock is not allowed.

NOTE Plan ahead for the future growth of the cluster. If the cluster will grow to more than 4 nodes, you should either not configure a cluster lock or else plan on taking down the cluster while the fifth node is added so the cluster lock can be removed.

Single Cluster Lock

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

Dual Cluster Lock

If you are using disks that are internally mounted in the same cabinet as the cluster nodes, then a single lock disk would be a single point of failure in this type of cluster, since the loss of power to the node that has the lock disk in its cabinet would also render the cluster lock unavailable. In this case only, a dual cluster lock, with two separately powered cluster disks, should be used to eliminate the lock disk as a single point of failure. For a dual cluster lock, the disks must not share either a power circuit or a node chassis with one another. In this case, if there is a power failure affecting one node and disk, the other node and disk remain available, so cluster re-formation can take place on the remaining node.

No Cluster Lock

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

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

Backing Up Cluster Lock 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 whether you use SAM or HP-UX commands to create the lock volume group.

How the Package Manager Works

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

The package coordinator does the following:

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

The package manager on all nodes does the following:

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

Deciding When and Where to Run and Halt Packages

Each package is separately configured by means of a package configuration file, which can be edited manually or through SAM. This file assigns a name to the package and identifies the nodes on which the package can run, in order of priority. It also indicates whether or not switching is enabled for the package, that is, whether the package should switch to another node or not in the case of a failure. There may be many applications in a package. Package configuration is described in detail in the chapter "Configuring Packages and their Services."

Starting the Package and Running Application Services

After a cluster has formed, the package manager on each node starts up packages on that node. Starting a package means running individual application services on the node where the package is running.

To start a package, the package manager runs the **package control script** with the 'start' parameter. This script performs the following tasks:

- uses Logical Volume Manager (LVM) commands to activate volume groups needed by the package.
- mounts filesystems from the activated volume groups to the local node.
- uses `cmmodnet` to add the package's IP address to the current network interface running on a configured subnet. This allows clients to connect to the same address regardless of the node the service is running on.

- uses the `cmrunserv` command to start up each application service configured in the package. This command also initiates monitoring of the service.
- executes a set of customer-defined run commands to do additional processing, as required..

While the package is running, services are continuously monitored. If any part of a package fails, the package halt instructions are executed as part of a recovery process. Failure may result in simple loss of the service, a restart of the service, transfer of the package to an adoptive node, or transfer of all packages to adoptive nodes, depending on the package configuration. In package transfers, MC/ServiceGuard sends a TCP/IP packet across the heartbeat subnet to the package's adoptive node telling it to start up the package.

NOTE

When applications run as services in an MC/ServiceGuard package, you do not start them directly; instead, the package manager runs packages on your behalf either when the cluster starts or when a package is enabled on a specific node. Similarly, you do not halt an individual application or service directly once it becomes part of a package. Instead you halt the package or the node. Refer the chapter on "Cluster and Package Maintenance" for a description of how to start and stop packages.

The package configuration file and control script are described in detail in the chapter "Configuring Packages and their Services."

Service Monitor

The Service Monitor checks the PIDs of services started by the package control script. If it detects a PID failure, the package is halted. Depending upon the parameters set in the package configuration file, MC/ServiceGuard will attempt to restart the service on the primary node, or the package will fail over to a specified adoptive node.

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 MC/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 MC/ServiceGuard. Monitors currently supplied with other software products include an OTS/9000 monitor and an ATM monitor.

If a registered resource is configured in a package, the package manager calls the resource registrar to launch an external monitor for the resource. The monitor then sends messages back to MC/ServiceGuard, which can fail the package to another node or take other action if the resource is considered unavailable.

Using the EMS HA Monitors

The EMS HA Monitors, available as a separate product (A5735AA), can be used to set up monitoring of disks and other resources as package dependencies. Examples of resource attributes that can be monitored using EMS include the following:

- Logical volume status
- Physical volume status
- System load
- LAN health

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

The EMS HA Monitors can also be used to report monitored events to a target application such as ClusterView for graphical display or for operator notification. Refer to the manual *Using EMS HA Monitors* (HP part number B5735-90001) for additional information.

Stopping the Package

The package manager is notified when a command is issued to shut down a package. In this case, the package control script is run with the 'stop' parameter. For example, if the system administrator chooses "Halt Package" from the "Package Administration" menu in SAM, the package manager will stop the package. Similarly, when a command is issued to halt a cluster node, the package manager will shut down all the packages running on the node, executing each package control script with the 'stop' parameter. When run with the 'stop' parameter, the control script:

- uses `cmhaltserv` to halt each service.
- unmounts filesystems that had been mounted by the package.
- uses Logical Volume Manager (LVM) commands to deactivate volume groups used by the package.

- uses `cmmodnet` to delete the package's IP address from the current network interface.

NOTE The package is automatically stopped on the failure of a package component. For more details, refer to "Responses to Package and Service Failures," below.

How the Network Manager Works

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

Stationary and Relocatable IP Addresses

Each node (host system) should have an 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. A stationary IP address is not transferrable to another node, but may be transferrable to a standby LAN interface card. The stationary IP address is *not* associated with packages. Stationary IP addresses are used to transmit heartbeat messages (described earlier in the section "How the Cluster Manager Works") and other data.

In addition to the stationary IP address, you normally assign one or more unique IP addresses to each package. The package IP address is assigned to the primary LAN interface card by the `cmmodnet` command in the package control script when the package starts up. The IP addresses associated with a package are called **relocatable IP addresses** (also known as **package IP addresses** or **floating IP addresses**) because the addresses can actually move from one cluster node to another. You can use up to 200 relocatable IP addresses in a cluster spread over as many as 30 packages.

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

Both stationary and relocatable IP addresses will switch to a standby LAN interface in the event of a LAN card failure. In addition, relocatable addresses (but not stationary addresses) can be taken over by an adoptive node if control of the package is transferred. This means that applications can access the package via its relocatable address without knowing which node the package currently resides on.

Adding and Deleting Relocatable IP Addresses

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

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

Load Sharing

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

Monitoring LAN Interfaces and Detecting Failure

At regular intervals, MC/ServiceGuard polls all the network interface cards specified in the cluster configuration file. Network failures are detected in the following manner. One interface in a bridged net is assigned to be the poller. The poller will poll the other primary and standby interfaces in the bridged net 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.

The polling interface sends LAN packet messages to all other interfaces on a bridged net and receives packets back from all other interfaces on the bridged net. If an interface cannot receive or send a message, and when the numerical count of packets sent and received on an interface does not increment for an amount of time, the interface is considered DOWN.

Local Switching

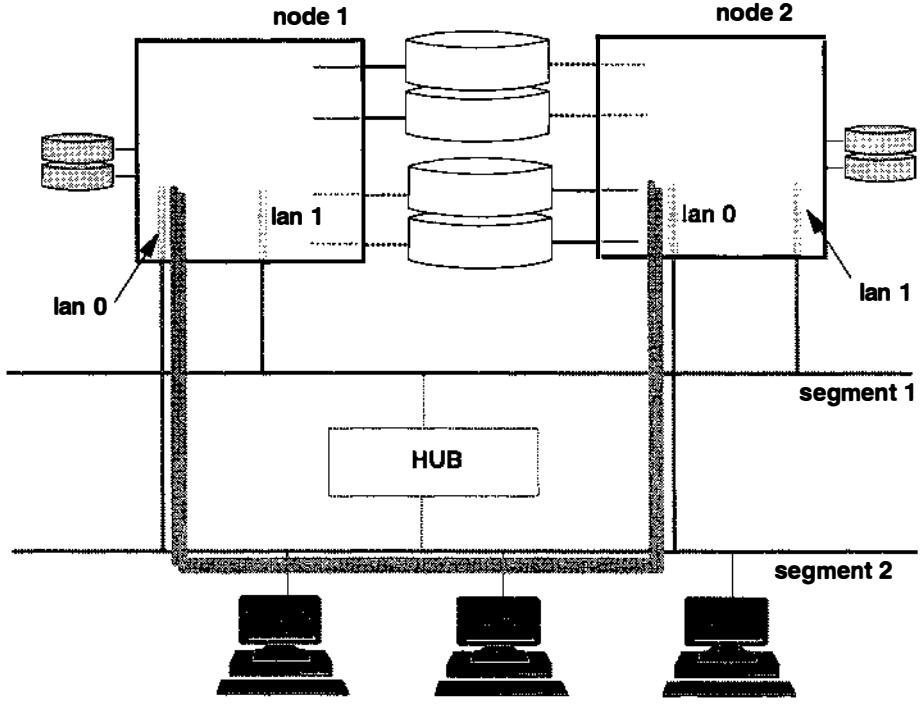
A local network switch involves the detection of a local network interface failure and a failover to the local backup LAN card. The backup LAN card must not have any IP addresses configured. In the case of a local network switch, TCP/IP connections are not lost for Ethernet, but IEEE 802.3 connections will be lost. Ethernet, Token Ring and FDDI use the ARP protocol, and HP-UX sends out an

unsolicited ARP to notify remote systems of address mapping between MAC (link level) addresses and IP level addresses. IEEE 802.3 does not have the *rearp* function.

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

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

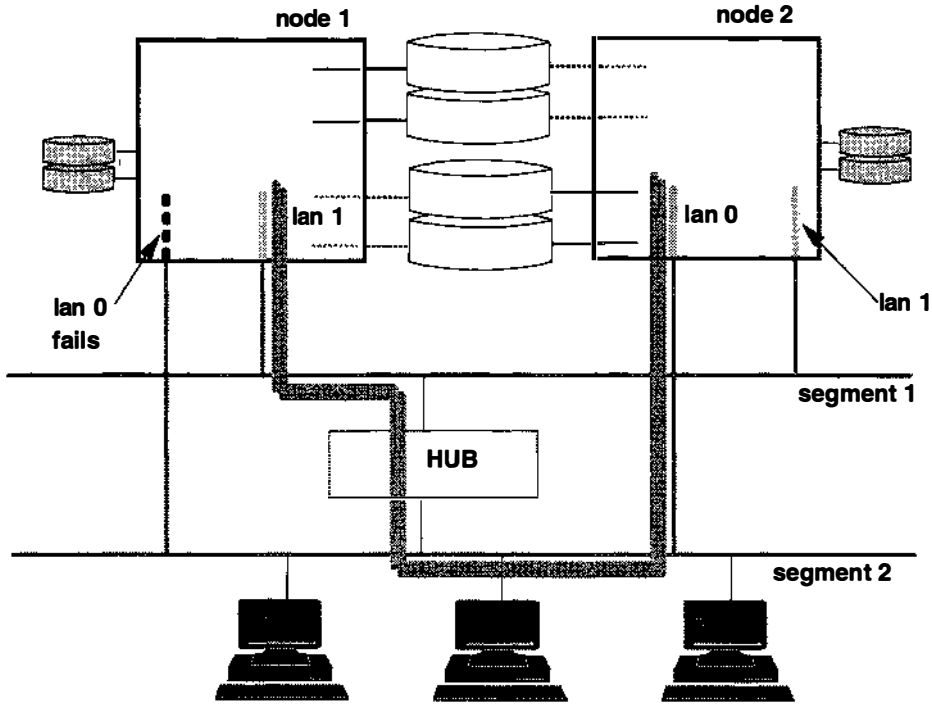
Figure 3-2 Cluster Before Local Network Switching



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

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

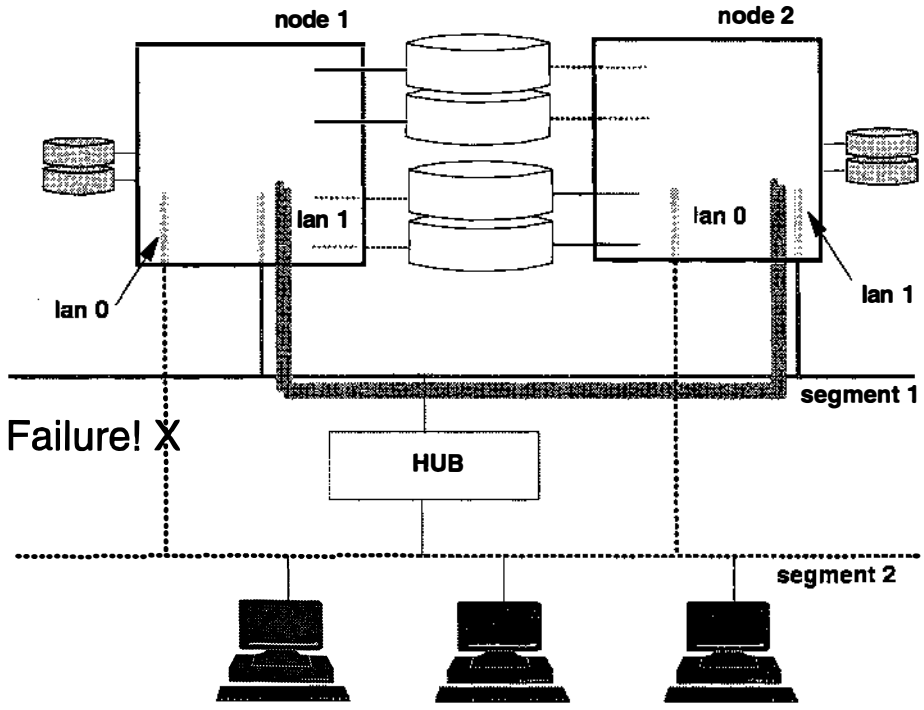
Figure 3-3 **Cluster After Local Network Switching**



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 2 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 2 is not affected.

Another example of local switching is shown in Figure 3-4. In this case a failure affecting segment 2 causes both nodes to switch to the LAN cards attached to segment 1.

Figure 3-4 **Local Switching After Cable Failure**



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

Remote Switching

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

The switching of relocatable IP addresses is shown in Figure 3-5 and Figure 3-6. Figure 3-5 shows a two node cluster in its original state with Package 1 running on Node 1 and Package 2 running on Node 2. Users connect to node with the IP address of the package they wish to use. Each node has a stationary IP address associated with it, and each package has an IP address associated with it.

Figure 3-5 **Before Package Switching**

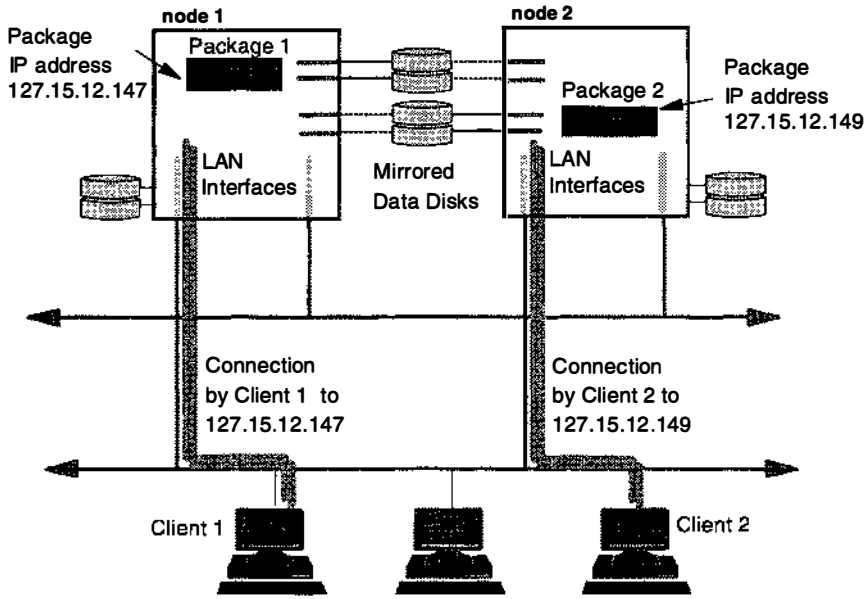
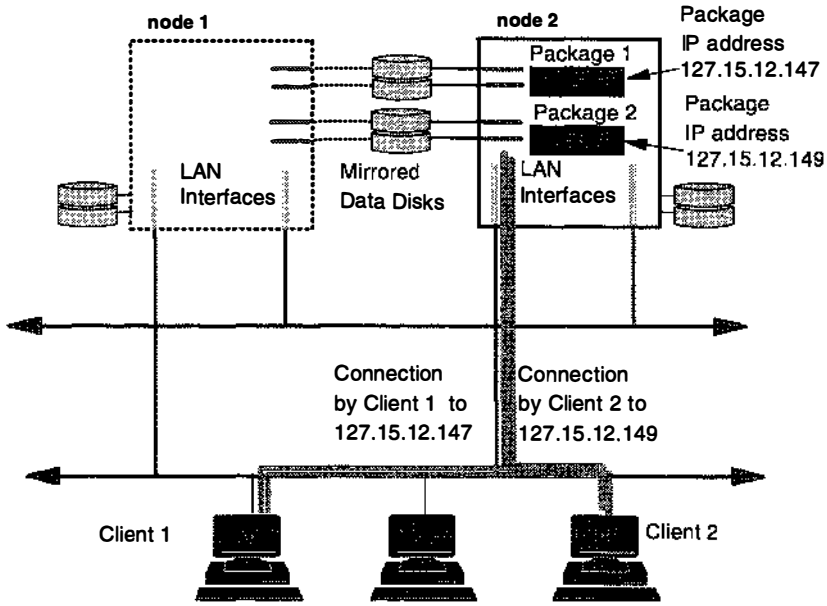


Figure 3-6 shows the condition where Node 1 has failed and Package 1 has been transferred to Node 2. Package 1's IP address was transferred to Node 2 along with the package. Package 1 continues to be available and is now running on Node 2. Also note that Node 2 can now access both Package A's disk and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

Figure 3-6 **After Package Switching**



ARP Messages after Switching

When a floating IP address is moved to a new interface, either locally or remotely, an ARP message is broadcast to indicate the new mapping between IP address and link layer address. An ARP message is sent for each IP address that has been moved. All systems receiving the broadcast should update the associated ARP cache entry to reflect the change.

Currently, the ARP messages are sent at the time the IP address is added to the new system. An ARP message is sent in the form of an ARP request. The sender and receiver protocol address fields of the ARP request message are both set to the same floating IP address. This ensures that nodes receiving the message will not send replies.

Responses to Failures

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

Transfer of Control (TOC) When a Node Fails

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

A TOC is done if there is a kernel hang, a kernel spin, a runaway real-time process, or if the MC/ServiceGuard daemon, *cmcld*, fails. During this event, a system dump is performed and the following message is sent to the console:

```
MC/ServiceGuard: Unable to maintain contact with cmcld daemon.
Performing TOC to ensure data integrity.
```

A TOC is also initiated by MC/ServiceGuard itself under specific circumstances. If the service failfast parameter is enabled in the package configuration file, the entire node will fail with a TOC whenever there is a failure of that specific service. If the package failfast parameter is enabled in the package configuration file, the entire node will fail with a TOC whenever there is a failure causing the package control script to exit. In addition, a node-level failure may also be caused by events independent of a package and its services. Loss of the heartbeat or loss of the MC/ServiceGuard or other critical daemons will cause a node to fail even when its packages and their services are functioning.

Responses to Hardware Failures

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

In the event of a LAN interface failure, a local switch is done to a standby LAN interface if one exists. If a heartbeat LAN interface fails and no standby is configured, the node fails with a TOC. If a data LAN interface fails without a standby, the node fails with a TOC only if *Package Failfast* (described further in the "Planning" chapter under "Package Configuration Planning") is enabled for the package.

Disk protection is provided by the separate product MirrorDisk/UX. In addition, separately available EMS disk monitors allow you to notify operations personnel when a failure takes place. Refer to the manual *Using the Event Monitoring Service* (HP part number B5735-90001) for additional information.

MC/ServiceGuard does not respond directly to power failures, although a loss of power to an individual cluster component may appear to MC/ServiceGuard like the failure of that component, and will result in the appropriate switching behavior.

Power protection is provided by PowerTrust, HP's uninterruptible power supply.

Responses to Package and Service Failures

In the default case, the failure of the package or of a service within a package causes the package to shut down by running the control script with the 'stop' parameter, and then restarting the package on an alternate node. If the package manager receives a report of an EMS monitor event showing that a resource is down, the package fails.

If you wish, you can modify this default behavior by specifying that the node should crash (TOC) before the transfer takes place. In cases where package shutdown may take a long time but the package is crash-tolerant and can recover quickly on restart, this option can make the package and its associated applications available to users more quickly. Remember, however, that when the node crashes, *all* packages on the node are halted abruptly.

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

Service Restarts

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

Network Communication Failure

An important element in the cluster is the health of the network itself. As it continuously monitors the cluster, each node listens for heartbeat messages from the other nodes confirming that all nodes are able to communicate with each other. If a node does not hear these messages within the configured amount of time, a node timeout occurs, resulting in a TOC. In a two-node cluster, the use of an RS-232 line prevents a TOC from the momentary loss of heartbeat on the LAN due to network saturation. The RS232 line also assists in quickly detecting network failures when they occur.

4 Planning and Documenting an HA Cluster

Building an MC/ServiceGuard cluster begins with a planning phase in which you gather and record information about all the hardware and software components of the configuration. Planning begins with a simple list of hardware and network components. As the installation and configuration continue, the list is extended and refined. After hardware installation, you can use the SAM high availability options or a variety of HP-UX commands to obtain information about your configuration; this information is entered on the worksheets provided in this chapter. During the creation of the cluster, the planning worksheets provide the values that are input with SAM or edited into the configuration files and control scripts.

This chapter assists you in the following planning areas:

- General Planning
- Hardware Planning
- Power Supply Planning
- Volume Group and Physical Volume Planning
- Cluster Configuration Planning
- Package Configuration Planning

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

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

Subsequent chapters describe configuration and maintenance tasks in detail.

General Planning

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

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

Planning for Expansion

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

- Set the Maximum Configured Packages parameter (described later in this chapter in the "Cluster Configuration Planning" section) high enough to accomodate the additional packages you plan to add. Keep in mind that adding package capacity uses memory resources (600K per package).

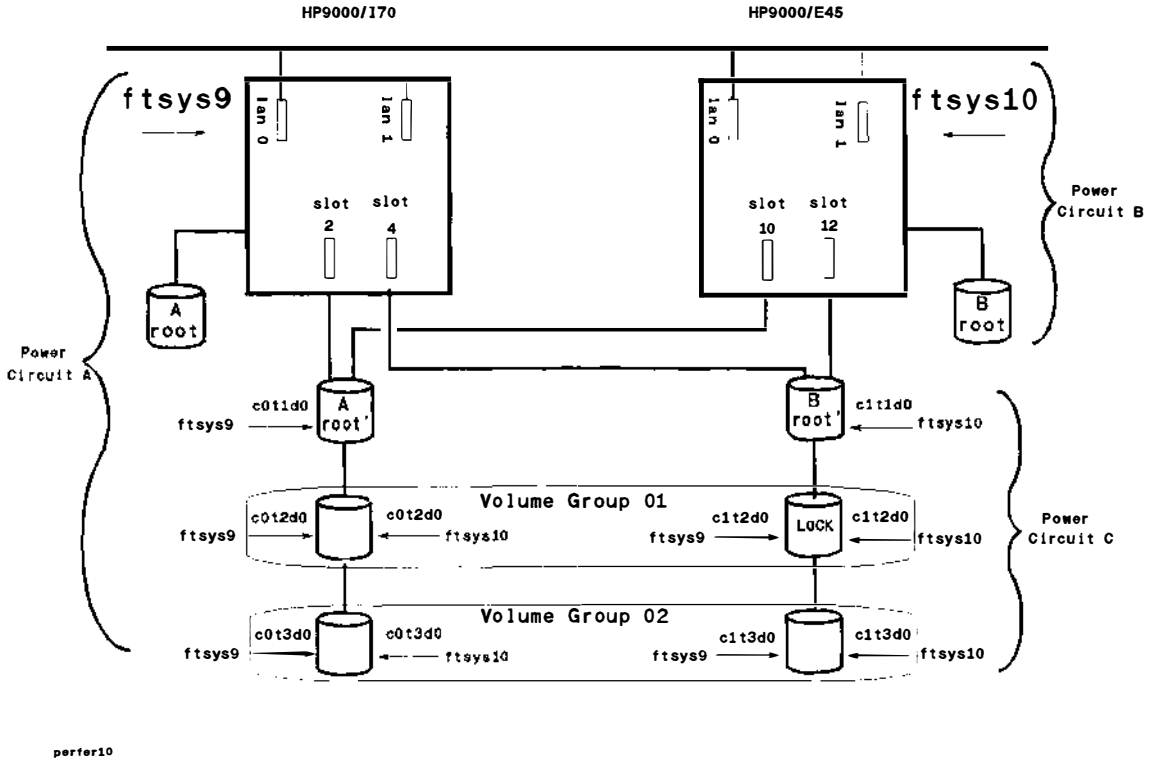
- Plan SCSI bus cabling to allow the addition of more disk hardware for shared data storage if needed. You should attach inline SCSI terminator cables to SCSI ports that you intend to use for additional devices. This allows you to add them while the bus is still active.
- Remember the rules for cluster locks when considering expansion. A cluster of two nodes must have a cluster lock, while a cluster of five or more nodes may *not* have a cluster lock. This means that if you intend to expand to a size greater than four nodes while the cluster is running, you should begin with at least three nodes and not configure a cluster lock.
- Networks should be pre-configured into the cluster configuration if they will be needed for packages you will add later while the cluster is running.
- Resources monitored by EMS should be pre-configured into the cluster configuration if they will be needed for packages you will add later while the cluster is running. Once a resource dependency is configured for any package in the cluster, it will always be available for later packages to use. However, you cannot add a never-before-configured resource to a package while the cluster is running.

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

Hardware Planning

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

Figure 4-1 Sample Cluster Configuration



Create a similar sketch for your own cluster, and record the information on the Hardware Worksheet. Indicate which device adapters occupy which slots, and determine the bus address for each adapter. Update the details as you do the cluster configuration (described in Chapter 3). *Use one form for each SPU*. The form has three parts:

- SPU Information

- Network Information
- Disk I/O Information

SPU Information

All HP 9000 Series 800 HP-PA SPUs are supported by MC/ServiceGuard and different models can be mixed in the same cluster. This includes both uniprocessor and multiprocessor computers. HP 9000 Series 700 SPUs are not supported by MC/ServiceGuard. SPU information includes the basic characteristics of the S800 systems you are using in the cluster. On the worksheet, include the following items:

S800 Series Number — Enter the series number, e.g., T600.

Host Name — Enter the name to be used on the system as the host name.

Memory Capacity — Enter the memory in MB.

Number of I/O slots — Indicate the number of slots.

NFS diskless clusters and NetLS servers are not supported.

Network Information

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

LAN Information

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

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

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

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

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

Kind of LAN Traffic
Identify the purpose of the subnet. Valid types include the following:

- Heartbeat
- Client Traffic
- Standby

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

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

MC/ServiceGuard supports the following local area networks:

- Ethernet
- Token Ring
- 100VG
- 10BaseT
- 100BaseT
- IEEE 802.3
- FDDI
- Fibre Channel (EPS clusters only)

For Ethernet or IEEE 802.3 configurations, subnets must be either all Ethernet or all IEEE 802.3, respectively.

MC/ServiceGuard does not directly support:

- NFS diskless clusters
- NetLS applications that use hostname or rely on the routine `gethostname()`,

- Local switching for applications that use link-level addressing.

RS232 Information

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

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

   ```
   # ioscan -fnC tty
   ```

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

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

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

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

Node Name	Name of the node
RS232 Device File	Enter the device file name corresponding to a serial interface on each node. This parameter is known as `SERIAL_DEVICE_FILE` in the ASCII configuration file.

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

Table 4-1 **Fast/Wide SCSI Addressing in Cluster Configuration**

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

Disk I/O Information

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

Bus Type — Indicate the type of bus. Supported busses are single-ended SCSI, F/W SCSI and FL (fiber link).

Slot Number — Indicate the slot number in which the card is inserted. For both F/W SCSI and FL disks, use the even number printed at the bottom of the slot. (Not relevant for D and K systems.)

Address — Enter the bus hardware path number, which will be seen on the system later when you use `ioscan` to display hardware. The address is given by the formula 4*(slot number). E.g., for slot number 4, the hardware path number is 16. (Not relevant for D and K systems.)

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

Information from this section of the worksheet is used in creating the mirrored disk configuration using Logical Volume Manager. In addition, it is useful to gather as much information as possible about your disk configuration. You can obtain information about available disks by using the following commands:

- `diskinfo`
- `ioscan -fnC` *disk*
- `lssf /dev/dsk/c*`
- `bdf`
- `mount`
- `swapinfo`
- `vgdisplay -v`
- `lvdisplay -v`
- `lssf /dev/*dsk/*d0`

These are standard HP-UX commands. See their man pages for information of specific usage.

The commands should be issued from *all nodes* after installing the hardware and rebooting the system. The information will be useful when doing LVM and cluster configuration. The output from the `lssf` command can be marked to indicate which physical volume group a disk is assigned to.

Hardware Configuration Worksheet

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

```
SPU Information:

S800 Host Name ____ftsys9_____    S800 Series No _____892_____

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 __Blude__ Interface ___lan2___ Addr_____________ Type   _standby___

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

Serial (RS232) Heartbeat Interface Information:

Node Name ____node1_____ RS232 Device File ___/dev/tty0p0__

Node Name ____node2_____ RS232 Device File ___/dev/tty0p0__

X.25 Information

OTS subnet  _____________________ OTS subnet  _____________________

Disk I/O Information for Shared Disks:

Bus Type _SCSI_ Slot Number _4__ Address _16_  Disk Device File /dev/dsk/c0t1d0

Bus Type _SCSI_ Slot Number _6_  Address _24_  Disk Device File  /dev/dsk/c0t2d0

Bus Type _____ Slot Number ___  Address ____  Disk Device File  _____________

Bus Type _____ Slot Number ___  Address ____  Disk Device File  _____________


Attach a printout of the output from 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. No more than half of the nodes should be on a single power source. If a power source supplies exactly half of the nodes, it must not also supply the cluster lock disk or the cluster will not be able to reform after a failure. See the section on cluster locks in "Cluster Configuration Planning" for more information.

To provide a high degree of availability in the event of power failure, use a separate UPS at least for each node's SPU and for the cluster lock disk, and ensure that disks which will be configured as members of separate physical volume groups are connected to different power supplies. This last rule enables disk mirroring to take place between physical disks that are connected to different power supplies as well as being on different I/O busses.

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

S800 Host Name	Enter the host name for each SPU.
Disk Unit	Enter the disk drive unit number for each disk.
Tape Unit	Enter the tape unit number for each backup device.
Other Unit	Enter the number of any other unit.
Power Supply	Enter the power supply unit number of the UPS to which the host or other device is connected.

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

NOTE Battery backup functionality on the Series 890 system is not supported by MC/ServiceGuard.

Power Supply Configuration Worksheet

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

```
SPU Power:

S800 Host Name ____ftsys9_____     Power Supply _____1_____

S800 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 _____
```

Volume Group and Physical Volume Planning

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

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

If you plan to use the EMS HA Disk Monitor, refer to the section on "Rules for Using EMS Disk Monitor with MC/ServiceGuard" in the manual *Using EMS HA Monitors* (B5735-90001).

Volume Group and Physical Volume Worksheet

The following worksheet will help you organize and record your specific physical disk configuration. Make as many copies as you need. Fill out the worksheet and keep it for future reference. If you are using disk arrays, use the worksheet in the next section, "Volume Group and PV Links Worksheet."

It is recommended that you use volume group names other than the default volume group names (vg01, vg02, etc.). Choosing volume group names that represent the high availability applications that they are associated with (for example, `/dev/vgdatabase` will simplify cluster administration).

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

Volume Group and PV Links Worksheet

The following worksheet will help you organize and record your specific physical disk configuration if you are using PV (alternate) links with disk arrays. Make as many copies as you need. Fill out the worksheet and keep it for future reference.

```
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:  _____
```

Cluster Configuration Planning

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

- The length of the cluster heartbeat interval and node timeout. They should each be set as short as practical, but not shorter than 1000000 (one second) and 2000000 (two seconds), respectively.
- The design of the run and halt instructions in the package control script. They should be written for fast execution.
- The availability of raw disk access. Applications that use raw disk access should be designed with crash recovery services.
- The application and database recovery time. They should be designed for the shortest recovery time.

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

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

Choosing the Cluster Lock Volume Group

A specific disk is identified in the cluster configuration file as holding the cluster lock. This disk must be accessible from all nodes in the cluster. 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.

Cluster Lock and Re-formation Time

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

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

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

The following list shows disk interface types in descending order of cluster lock disk acquisition/re-formation time:

- Any combination of HP-HSC and HP-PB disks on F/W SCSI
- Fiber Link disks
- HP-PB disks on single-ended SCSI

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

Heartbeat Subnet and Re-formation Time

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

Planning for Expansion

You can add additional cluster nodes after the cluster is up and running, but doing so without bringing down the cluster requires you to follow some rules. Recall that a cluster with more than 4 nodes may not have a cluster lock. Thus, if you plan to add enough nodes to bring the total to more than 4, you must start with 3 nodes, since a two-node cluster requires a cluster lock. Also, new nodes added to the cluster must be on the same subnet as the other cluster nodes.

Cluster Manager Parameters

For the operation of the cluster manager, you need to define a set of cluster parameters. These are stored in the binary cluster configuration file, which is located on all nodes in the cluster. These parameters can be entered by using SAM or by

editing the cluster configuration template file created by issuing the `cmquerycl` command, as described in the chapter "Building an HA Cluster Configuration." The parameter names given below are the names that appear in SAM. The names coded in the ASCII cluster configuration file appear at the end of each entry.

The following parameters must be identified:

Cluster Name
: The name of the cluster as it will appear in the output of `cmviewcl` and other commands, and as it appears in the cluster configuration file.

  In the ASCII cluster configuration file, this parameter is `CLUSTER_NAME`.

  The cluster name must not contain any of the following illegal characters: '/', '\', and '*'. All other characters are legal.

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

  In the ASCII cluster configuration file, this parameter is `NODE_NAME`.

Cluster Aware Volume Group
: The name of a volume group whose disks are attached to at least two nodes in the cluster. Such disks are considered cluster aware.

  In the ASCII cluster configuration file, this parameter is `VOLUME_GROUP`.

Heartbeat Subnet
: IP notation in SAM indicating the subnet that will carry the cluster heartbeat. Note that heartbeat IP addresses must be on the same subnet on each node.

  In the ASCII cluster configuration file, this parameter is `HEARTBEAT_IP`.

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

  In the ASCII cluster configuration file, this parameter is `SERIAL_DEVICE_FILE`.

Monitored Non-Heartbeat Subnet

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

In the ASCII cluster configuration file, this parameter is `STATIONARY_IP`.

Lock Volume Group

The volume group containing the physical disk volume on which a cluster lock is written. Identifying a cluster lock volume group is essential in a two-node cluster. If you are creating two cluster locks, enter the volume group name or names for both locks.

In the ASCII cluster configuration file, this parameter is `FIRST_CLUSTER_LOCK_VG` for the first lock volume group. If there is a second lock volume group, the parameter `SECOND_CLUSTER_LOCK_VG` is included in the file on a separate line.

Physical Volumes

The name of the physical volume within the Lock Volume Group that will have the cluster lock written on it. Enter the physical volume name as it appears on both nodes in the cluster (the same physical volume may have a different name on each node). If you are creating two cluster locks, enter the physical volume names for both locks.

In the ASCII cluster configuration file, this parameter is `FIRST_CLUSTER_LOCK_PV` for the first physical lock volume and `SECOND_CLUSTER_LOCK_PV` for the second physical lock volume. If there is a second physical lock volume, the parameter `SECOND_CLUSTER_LOCK_PV` is included in the file on a separate line.

Disk Unit

This information is for a label to be attached to each disk drive. Enter the number of the disk drive unit on which the physical volume is located.

Power Supply

This information is for a label to be attached to each UPS. Enter the number of the power supply to which the physical volume is connected.

Heartbeat Interval The normal interval between the transmission of heartbeat messages from one node to the other in the cluster. Enter a number of seconds. Default: 1 second. The interval should not be set smaller than this.

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

Node Timeout The time after which a node may decide that the other node has become unavailable and initiate reconfiguration. Enter a number of seconds. Default: 2 seconds. Minimum is 2 * (Heartbeat Interval).

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

Maximum Configured Packages This parameter sets the maximum number of packages that can be configured in the cluster. The default is 0, which means that you must set this parameter if you want to use packages. The greatest possible value is 30.

Set this parameter to a value that is high enough to accomodate a reasonable amount of future package additions without the need to bring down the cluster to reset the parameter. However, be sure not to set the parameter so high that memory is wasted. Each configured package requires about 600 K of lockable memory.

In the ASCII cluster configuration file, this parameter is known as `MAX_CONFIGURED_PACKAGES`.

Network Polling Interval The frequency at which the networks configured for MC/ServiceGuard are checked. The current default is 2 seconds. Thus every 2 seconds, the cluster manager polls each network interface to make sure it can still send and receive information. Changing this value can affect how quickly a network failure is detected.

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

Autostart Delay 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 number of seconds 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: 600 seconds.

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

Cluster Configuration Worksheet

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

```
Name and Nodes:

Cluster Name: ____cluster1_____

Node Names: ______ftsys9, ftsys10_____

Maximum Configured Packages: _______6_____

Cluster Volume Groups: ___/dev/vg01, /dev/vg02_____

Subnets:

Heartbeat IP Addresses: ____15.13.171.32 and 192.6.7.3_____

Non-Heartbeat IP Addresses: __192.6.143.10_____

Cluster Lock Volume Groups and Volumes:

First Lock VG:          First Lock Physical Volume:

  ___/dev/vg01_____|Name (Node 1): /dev/dsk/c1t2d0__Disk Unit No: __Power Unit: ___
                   |
                   |Name (Node 2): /dev/dsk/c1t2d0__Disk Unit No: __Power Unit: ___
                   |
Second Lock VG:    |  Second Lock Physical Volume:
                   |
  _____ _____      |Name (Node 1): _____Disk Unit No: __Power Unit: ___
                   |
                   |Name (Node 2): _____Disk Unit No: __Power Unit: ___

Timing Parameters:

Heartbeat Interval: __1 sec__

Node Timeout: __2 sec_

Network Polling Interval: _2 sec_

Autostart Delay: __10 min__
```

Package Configuration Planning

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

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

Logical Volume and Filesystem Planning

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

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

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

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

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

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

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

```
# /dev/vg01/lvoldb1 /applic1 vxfs defaults 0 1   # These six 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 MC/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.

CAUTION

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

Details about creating, exporting, and importing volume groups in MC/ServiceGuard are given in the chapter on "Building an HA Cluster Configuration."

Monitoring Registered Package Resources

MC/ServiceGuard has access to a registry of resources that can be monitored as package dependencies. The registry is the core of the Event Monitoring Service (EMS). Once an EMS registered resource is configured as a package dependency, MC/ServiceGuard can fail a package to another node based on messages the resource's monitor returns. Monitors for individual resources may be provided by hardware or software vendors from time to time. A specific group of HA EMS monitors for disk, LAN, and system status information is available from HP as a separate product. Refer to the manual *Using EMS HA Monitors* (B5735-90001) for additional information.

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

Choosing Switching and Failover Behavior

Switching IP addresses from one LAN card to another may take place if Automatic Switching is set to Enabled in SAM (`NET_SWITCHING_ENABLED` set to YES in the ASCII package configuration file). Automatic Switching Enabled is the default. The following table describes different types of failover behavior and the settings that determine each behavior in SAM or in the ASCII package configuration file.

Table 4-2 Package Failover Behavior

Switching Behavior	Options in SAM	Parameters in ASCII File
Package IP address switches to standby LAN card transparently on LAN card failure	• Automatic Switching set to Enabled for the package (Default)	• `NET_SWITCHING_ENABLED` set to YES for the package (Default)
Package switches normally after detection of failure or report of an EMS monitor event showing that a resource on which the package depends is down. Halt script runs before switch takes place (default behavior.)	• Package Failfast set to Disabled. (Default) • Service Failfast set to Disabled for all services. (Default) • Automatic Switching set to Enabled for the package. (Default)	• `NODE_FAIL_FAST_ENABLED` set to NO. (Default) • `SERVICE_FAIL_FAST_ENABLED` set to NO for all services. (Default) • `PKG_SWITCHING_ENABLED` set to YES for the package. (Default)

Switching Behavior	Options in SAM	Parameters in ASCII File
All packages switch following a TOC (Transfer of Control, an immediate halt without a graceful shutdown) on the node when a specific service fails. Halt scripts are not run.	• Package Failfast set to Disabled • Service Failfast set to Enabled for a specific service • Automatic Switching set to Enabled for all packages.	• NODE_FAIL_FAST_ENABLED set to NO • SERVICE_FAIL_FAST_ENABLED set to YES for a specific service. • PKG_SWITCHING_ENABLED set to YES for all packages.
All packages switch following a TOC on the node when any service fails.	• Package Failfast set to Disabled. • Service Failfast set to Enabled for *all* services. • Automatic Switching set to Enabled for all packages.	• NODE_FAIL_FAST_ENABLED set to NO. • SERVICE_FAIL_FAST_ENABLED set to YES for *all* services. • PKG_SWITCHING_ENABLED set to YES for all packages.
All packages switch following a TOC on the node when the run or halt script exits with an error other than 0 or 1. This may be caused by an EMS monitor event showing that a resource is down	• Package Failfast set to Enabled. • Automatic Switching set to Enabled for all packages.	• NODE_FAIL_FAST_ENABLED set to YES. • PKG_SWITCHING_ENABLED set to YES for all packages.

Planning for Expansion

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

Package Configuration File Parameters

Prior to generation of the package configuration file, assemble the following package configuration data. The parameter names given below are the names that appear in SAM. The names coded in the ASCII cluster configuration file appear at the end of each entry. The following parameters must be identified and entered on the worksheet *for each package*:

Package Name
The name of the package. The package name must be unique in the cluster. It is used to start, stop, modify, and view the package.

The package name must not contain any of the following illegal characters: '/', '\', and '*'. All other characters are legal.

Node Name
The names of primary and alternate nodes for the package, e.g., ftsys9 and ftsys10. The order in which you specify the node names is important. First list the primary node name, then the first adoptive node name, then the second adoptive node name, followed, in order, by additional node names. Transfer of control of the package will occur to the next adoptive node name listed in the package configuration file.

Control Script Pathname
Enter the full pathname of the package control script. (The script must reside in a directory that contains the string "`cmcluster`".) It is recommended that you use the same script as both the run and halt script. This script will contain both your package run instructions and your package halt instructions. When the package starts, its run script is executed and passed the parameter 'start'; similarly, at package halt time, the halt script is executed and passed the parameter 'stop'.

In the ASCII package configuration file, this parameter maps to the two separate parameters named `RUN_SCRIPT` and `HALT_SCRIPT`. Use the name of the single control script as the name of the `RUN_SCRIPT` and the `HALT_SCRIPT` in the ASCII file.

If you wish to separate the package run instructions and package halt instructions into separate scripts, the package configuration file allows allows you to do this by naming two separate scripts. However, under most conditions, it is simpler

to combine your run and halt instructions into a single package control script and repeat its name for both the RUN_SCRIPT and the HALT_SCRIPT.

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

Ensure that this script is executable.

Run Script Timeout and Halt Script Timeout

Enter a number of seconds. If the script has not completed by the specified timeout value, MC/ServiceGuard will terminate the script. The default is 0, or no timeout.

If the timeout is exceeded:

- Control of the package will not be transferred.
- The run or halt instructions will not be run.
- Global switching will be disabled.
- The current node will be disabled from running the package.
- The control script will exit with status 1.

In the ASCII package configuration file, this parameter is called RUN_SCRIPT_TIMEOUT and HALT_SCRIPT_TIMEOUT. The default for both is 0 or NO_TIMEOUT. In the ASCII file, this parameter is entered in microseconds.

If the halt script timeout occurs, you may need to perform manual cleanup See "Package Control Script Hangs or Failures" in the "Troubleshooting" chapter.

Service Name

Enter a unique name for each service. You can configure a maximum of 30 services per package.

In the ASCII package configuration file, this parameter is called SERVICE_NAME. Define one SERVICE_NAME entry for each service.

Service Fail Fast

Enter Enabled or Disabled for each service. This parameter indicates whether or not the failure of a service results in the failure of a node. If the parameter is set to Enabled, in the event

of a service failure, MC/ServiceGuard will halt the node on which the service is running with a TOC. The default is Disabled.

In the ASCII package configuration file, this parameter is SERVICE_FAIL_FAST_ENABLED, and possible values are YES and NO. The default is NO. Define one SERVICE_FAIL_FAST_ENABLED entry for each service.

The service name must not contain any of the following illegal characters: '/', '\', and '*'. All other characters are legal.

Service Halt Timeout

In the event of a service halt, MC/ServiceGuard will first send out a SIGTERM signal to terminate the service. If the process is not terminated, MC/ServiceGuard will wait for the specified timeout before sending out the SIGKILL signal to force process termination. Default is 300 seconds (5 minutes).

In the ASCII package configuration file, this parameter is SERVICE_HALT_TIMEOUT. Define one SERVICE_HALT_TIMEOUT entry for each service.

Subnet

Enter the IP subnets that are to be monitored for the package.

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

Resource Name

The name of a resource that is to be monitored by MC/ServiceGuard as a package dependency. A resource name is the name of an important attribute of a particular system resource. The resource name includes the entire hierarchy of resource class and subclass within which the resource exists on a system.

In the ASCII package configuration file, this parameter is called RESOURCE_NAME. Obtain the resource name from the list provided in SAM, or obtain it from the documentation supplied with the resource monitor.

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

Resource Polling Interval

The frequency of monitoring an additional package resource. The default is 60 seconds. In the ASCII package configuration file, this parameter is called

RESOURCE_POLLING_INTERVAL. The Resource Polling Interval appears on the list provided in SAM, or you can obtain it from the documentation supplied with the resource monitor.

Resource Up Value The criteria for judging whether an additional package resource has failed or not. In the ASCII package configuration file, this parameter is called RESOURCE_UP_VALUE. The Resource Up Value appears on the list provided in SAM, or you can obtain it from the documentation supplied with the resource monitor.

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

Automatic Switching Enter Enabled or Disabled. The default is Enabled, which allows a package to start up normally on a cluster node. In the event of a failure, a value of Enabled permits MC/ServiceGuard to transfer the package to an adoptive node. If this parameter is set to Disabled, the package will not start up automatically when the cluster starts running.

In the ASCII package configuration file, this parameter is called PKG_SWITCHING_ENABLED, and possible values are YES and NO. The default is YES. If this parameter is set to NO, the package will not start up automatically when the cluster starts running.

Local Switching Enter Enabled or Disabled. In the event of a failure, this permits MC/ServiceGuard to switch LANs locally, that is, transfer to a standby LAN card. The default is Enabled.

In the ASCII package configuration file, this parameter is called NET_SWITCHING_ENABLED, and possible values are YES and NO. The default is YES.

Package Fail Fast Enabled In the event of the failure of the control script itself or the failure of a subnet or the report of an EMS monitor event showing that a resource is down, if this parameter is set to Enabled, MC/ServiceGuard will issue a TOC on the node where the control script fails. The default is Disabled.

In the ASCII package configuration file, this parameter is called NODE_FAIL_FAST_ENABLED, and possible values are YES and NO. The default is NO.

Package Control Script Variables

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

Volume Groups, Logical Volumes, File Systems and Mount Options

Determine the filesystems and corresponding logical volumes within the volume groups required. Example:

pkg1 requires /dev/vg01/lvol1 mounted on /vg01

Indicate the names of volume groups that are to be activated and deactivated, together with the logical volumes and file systems that are to be mounted. You can also specify options that are to be used with the HP-UX **mount** command. On starting the package, the script activates a volume group, and it may mount logical volumes onto file systems. At halt time, the script unmounts the file systems and deactivates each volume group. All volume groups must be accessible on each target node.

In the ASCII package control script, these variables are arrays, as follows: VG, LV, FS and FS_MOUNT_OPT. For each file system (FS), you must identify a logical volume (LV). Include as many volume groups (VGs) as needed. If you are using raw files, the LV, FS, and FS_MOUNT_OPT entries are not needed.

Only cluster aware volume groups should be specified in package control scripts. To make a volume group cluster aware, enter it as part of the cluster configuration. See above, "Cluster Configuration Planning."

IP Addresses and SUBNETs

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

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

Service Name Enter a unique name for each specific service within the package. All services are monitored by MC/ServiceGuard. The service name, service command, and service restart parameters are entered in the package control script in groups of three. You may specify as many service names as you need. Each name must be unique within the cluster. The service name is the name used by `cmrunserv` and `cmhaltserv` inside the package control script.

In the ASCII package control script, enter values into an array known as `SERVICE_NAME`. Enter one service name for each service.

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

In the ASCII package control script, enter values into an array known as `SERVICE_CMD`. Enter one service command string for each service.

Service Restart Parameter Enter a number of restarts. One valid form of the parameter is `-r` *n* where *n* is a number of retries. A value of "-r 0" indicates no retries. A value of "-R" indicates an infinite number of retries. The default is 0, or no restarts.

In the ASCII package control script, enter values into an array known as `SERVICE_RESTART`. Enter one restart value for each service.

For information on using a DTC with MC/ServiceGuard, see the chapter entitled "Configuring DTC Manager for Operation with MC/ServiceGuard" in the manual *Using the HP DTC Manager/UX*.

The package control script will clean up the environment and undo the operations in the event of an error.

Package Configuration Worksheet

Assemble your package configuration and control script data in a separate worksheet for each package.

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

Package Name: _____pkg1_____

Primary Node: _____ftsys9_____

First Failover Node:____ftsys10_____

Second Failover Node:_____

Package Run Script: __/etc/cmcluster/pkg1/control.sh__Timeout: _NO_TIMEOUT_

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

Package Switching Enabled?  __YES___ Local Switching Enabled?  ___YES__

Node Failfast Enabled?      ____NO____

Additional Package Resource:

Resource Name:_____ Polling Interval_____ Resource UP Value_____

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

Package Control Script Data:
================================================================================

VG[0]___/dev/vg01 __LV[0]__/dev/vg01/lvol1__FS[0]__/mnt1___FS_MOUNT_OPT[0]_____

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

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

IP[0] ___15.13.171.14 _____ SUBNET[0]_____15.13.168.0_____

IP[1] _____ SUBNET[1]_____

X.25 Resource Name _____

Service Name: __Svc1____ Run Command: __/usr/bin/MySvc -f_____Retries: _-r 2__

Service Fail Fast Enabled? ___NO___Service Halt Timeout __NO_TIMEOUT_____

Service Name: _____ Run Command: _____ Retries: _____

Service Fail Fast Enabled? _____Service Halt Timeout _____
```

5 Building an HA Cluster Configuration

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

- Preparing Your Systems
- Creating the Logical Volume Infrastructure
- Configuring the Cluster
- Verifying the Cluster Configuration
- Distributing the Binary Configuration File
- Preventing Automatic Activation of Volume Groups
- Setting up Autostart Features
- Changing the System Message
- Deleting the Cluster Configuration

Configuring packages is described in the next chapter.

Most configuration tasks can be completed using SAM (System Administration Manager) or using MC/ServiceGuard commands. You can use the SAM High Availability options to configure the cluster and packages as well as to start up the cluster on all nodes or on individual nodes. When you employ the SAM high availability options, you should be aware of the following user interface characteristics of SAM:

- SAM uses an object-action model. You select an object, then perform an action on it. The menu of actions available may vary depending on whether an object is selected or not.
- You must always deliberately select an item when choosing it from a list. Either click on the item or tab to it and press , then select **OK** to choose the item or items. An item is not selected by default even though it may appear to be highlighted.

- To make more than one selection from a list with a mouse, click on the first item with the left mouse button, then click on subsequent items by holding down **Ctrl** and pressing the left mouse button. Finally, select **OK** to choose the items.

The configuration and administration screens have extensive help associated with them.

If you are using MC/ServiceGuard commands to configure the cluster and packages, use the man pages for each command to obtain information about syntax and usage.

Preparing Your Systems

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

Editing Security Files

MC/ServiceGuard makes use of ARPA services to ensure secure communication among cluster nodes. Before installing MC/ServiceGuard, you must identify the nodes in the cluster that permit access by the root user on other nodes. If you do not do this, MC/ServiceGuard will not be able to copy files among nodes during configuration.

You can use SAM, or you can directly edit the `/.rhosts` file in the root home directory to include the names of all cluster nodes (including the node being used to configure the cluster) and the *root* user. The completed `/.rhosts` file will contain entries like the following:

```
node1 root
node2 root
```

where *node1* and *node2* are the names of the cluster nodes. The `/.rhosts` file should be copied to all cluster nodes.

You can also use the `/etc/hosts.equiv` and `/var/adm/inetd.sec` files to provide other levels of cluster security. For more information, refer to the HP 9000 guide, *Administering ARPA Services.*

Alternate Security File

Instead of using the `/.rhosts` file to enforce security in communication among all nodes within a cluster, MC/ServiceGuard allows you to specify an alternate file, `/etc/cmcluster/cmclnodelist`, for validating inter-node access within the cluster. MC/ServiceGuard will check for the existence of the `/etc/cmcluster/cmclnodelist` file first. If the `/etc/cmcluster/cmclnodelist` file exists, MC/ServiceGuard will only use this file to verify access within the cluster; otherwise, the `/.rhosts` file will be used. MC/ServiceGuard supports full domain names in both the `/etc/cmcluster/cmclnodelist` and `/.rhosts` files.

The format for entries in the /etc/cmcluster/cmclnodelist file is as follows:

[hostname] [rootuser] [#comment]

The following is an example:

```
nodeA                              root          # cluster1
nodeB.sys.dom.com                  root          # cluster1
```

Ensuring Consistency of Kernel Configuration

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

Enabling the Network Time Protocol

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

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

Preparing for Changes in Cluster Size

If you intend to add additional nodes to the cluster online, while it is running, ensure that they are connected to the same heartbeat subnets and to the same lock disks as the other cluster nodes. In selecting a cluster lock configuration, be careful to anticipate any potential need for additional cluster nodes. Remember that a cluster of more than four nodes *may not use* a cluster lock, but a two-node cluster *must* use a cluster lock. Thus, if you will eventually need five nodes, you should build an initial configuration with at least three to avoid the use of a cluster lock.

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

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

If you are planning to add a node online, and a package will run on the new node, ensure that any existing cluster bound volume groups for the package have been imported to the new node.

Creating the Logical Volume Infrastructure

Before 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. Separate procedures are given for the following:

- Mirrored Root
- Mirrored Individual Disks for Data
- Disk Arrays for Data

The Event Monitoring Service HA Disk Monitor provides the capability to monitor the health of LVM disks. If you intend to use this monitor for your mirrored disks, you should configure them in physical volume groups. For more information, refer to the manual *Using EMS HA Monitors* (B5735-90001).

Creating a Root Mirror

It is highly recommended that you use a mirrored root disk. Use the following procedure to create a LVM root mirror. This procedure cannot be carried out with SAM. In this example and in the following commands, *x* and *y* should be replaced with the corresponding numbers on the raw disk device file you are using.

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

   ```
   # pvcreate -B /dev/rdsk/cxtyd0
   ```

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

   ```
   # vgextend /dev/vg00 /dev/dsk/cxtyd0
   ```

3. Make the new disk a boot disk.

   ```
   # mkboot /dev/rdsk/cxtyd0
   ```

4. Copy the correct AUTO file into the new LIF area.

   ```
   # mkboot -a "hpux -l q (;0)/stand/vmunix" /dev/rdsk/cxtyd0
   ```

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

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

   ```
   # lvextend -m 1 /dev/vg00/lvol1 /dev/dsk/cxtyd0
   ```

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

   ```
   # lvextend -m 1 /dev/vg00/lvol2 /dev/dsk/cxtyd0
   ```

NOTE The root logical volume *must* be done first to ensure that it occupies the first contiguous set of extents on the new disk.

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

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

7. Check if the BDRA is correct.

```
# /usr/sbin/lvlnboot -R /dev/vg00
```

8. Verify that the mirror was properly created.

```
# lvlnboot -v
```

Creating Volume Groups for Mirrored Individual Data Disks

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

For more information on using LVM, refer to the chapter "Using HP-UX Commands to Manage Mirrors" in the "Logical Volume Manager" chapter of the *System Administration Tasks* manual for HP-UX 10.0 Series 800.

Using SAM to Create Volume Groups and Logical Volumes

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

1. Select the Volume Groups subarea.
2. From the Actions menu, choose Create or Extend.
3. Choose the first physical disk you wish to use in the volume group by selecting it from the list of available disks.
4. Enter the volume group name, e.g., `vgdatabase`.
5. Choose Create or Extend Volume Group.

6. Choose Add New Logical Volumes.
7. When adding logical volumes to the volume group, ensure that you are creating mirrored logical volumes with PVG strict allocation.
8. Specify the file system that is to be mounted on the volume group, for example, `/mnt1`.
9. Repeat the procedure for additional volume groups, logical volumes, and file systems.

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

Using LVM Commands to Create Volume Groups and Logical Volumes

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

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

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

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

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

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

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

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

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

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

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

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

   0xhh0000

   where *hh* must be unique to the volume group you are creating. Use the next hexadecimal number that is available on your system, after the volume groups that are already configured. 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 bus1 /dev/vgdatabase /dev/dsk/c1t2d0
   #vgextend -g bus2 /dev/vgdatabase /dev/dsk/c0t2d0
   ```

   The first command creates the volume group and adds a physical volume to it in a physical volume group called *bus1*. The second command adds the second drive to the volume group, locating it in a different physical volume group named *bus2*. The use of physical volume groups allows the use of PVG-strict mirroring of disks and PV links.

4. Repeat this procedure for additional volume groups.

Creating File Systems

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

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

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

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

2. Create a directory to mount the disk:

   ```
   #mkdir /mnt1
   ```

3. Mount the disk to verify your work:

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

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

4. Verify the configuration:

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

Creating Volume Groups on Disk Arrays Using PV Links

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

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

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

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

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

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

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

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

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

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

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

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

   0xhh0000

   where *hh* must be unique to the volume group you are creating. Use the next hexadecimal number that is available on your system, after the volume groups that are already configured. Use the following command to display a list of existing group files:

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

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

   ```
   # pvcreate /dev/dsk/c0t15d0
   ```

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

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

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

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

Creating Logical Volumes

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

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

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

NOTE

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

Deactivating the Volume Group

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

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

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

Distributing the Volume Group to Other Nodes

After creating volume groups for cluster data, you must make them available to any cluster node that will need to activate the volume group.

Distributing the Volume Group to Other Cluster Nodes with SAM

In SAM, choose the Disks and File Systems area, then choose Volume Groups. Select the volume group that is to be distributed to one or more additional nodes. Enter the name of each node that is to receive the volume group and select "Add." When the list is complete, press OK. SAM automatically configures the volume group for use on the other nodes.

After distributing the volume groups, check the /etc/lvmpvg files on each node to ensure that each physical volume group contains the correct physical volume names for that node.

Distributing the Volume Group to Other Cluster Nodes with LVM Commands

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

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

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

   ```
   # vgexport -p -m -s /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
   ```

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

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

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

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

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

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

NOTE

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

7. Enable the volume group on *ftsys10*:

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

8. Create a directory to mount the disk:

   ```
   # mkdir /mnt1
   ```

9. Mount and verify the volume group on ftsys10:

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

10. Unmount the volume group on *ftsys10*:

    ```
    # umount /mnt1
    ```

11. Deactivate the volume group on *ftsys10*:

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

Making Physical Volume Group Files Consistent

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

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

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

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

Creating Additional Volume Groups

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

Configuring the Cluster

This section describes how to define the basic cluster configuration. To do this in SAM, read the next section. If you want to use MC/ServiceGuard commands, skip ahead to the section entitled "Using MC/ServiceGuard Commands to Configure the Cluster."

Using SAM to Configure the Cluster

To configure a high availability cluster, use the following steps on the configuration node (ftsys9):

1. In SAM, select Clusters, then the High Availability Clusters option.
2. Choose the Cluster Configuration option. SAM displays a Cluster Configuration screen. If no clusters have yet been configured, the list area will be empty. If there are one or more HA clusters already configured on your local network, you will see them listed.
3. Select the Actions menu, and choose Create Cluster Configuration. A step menu appears.
4. Choose each required step in sequence, filling in the dialog boxes with required information, or accepting the default values shown. For information about each step, choose Help.
5. When finished with all steps, select **OK** at the Step Menu screen. This action creates the cluster configuration file and then copies the file to all the nodes in the cluster. When the file copying is finished, you return to the Cluster Configuration screen.
6. Exit from the Cluster Configuration screen, returning to the High Availability Clusters menu.

NOTE

In addition to creating and distributing a binary cluster configuration file, SAM creates an ASCII cluster configuration file, named /etc/cmcluster/cmclconfig.ascii. This file is available as a record of the choices entered in SAM.

Skip ahead to the section "Setting up Autostart Features."

Using MC/ServiceGuard Commands to Configure the Cluster

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

 # cmquerycl -v -C /etc/cmcluster/cmclconf.ascii -n ftsys9 -n ftsys10

The example creates an ASCII template file in the default cluster configuration directory, /etc/cmcluster. The ASCII file is partially filled in with the names and characteristics of cluster components on the two nodes *ftsys9* and *ftsys10*. Edit the filled-in cluster characteristics as needed to define the desired cluster. It is strongly recommended that you edit the file to send heartbeat over all possible networks, as shown in the following example.

Cluster Configuration Template File

```
# **********************************************************************
# ********* HIGH AVAILABILITY CLUSTER CONFIGURATION FILE ***************
# ***** For complete details about cluster parameters and how to    ****
# ***** set them, consult the cmquerycl(1m) manpage or your manual. ****
# **********************************************************************

# Enter a name for this cluster.  This name will be used to identify the
# cluster when viewing or manipulating it.

CLUSTER_NAME            cluster1


# Cluster Lock Device Parameters.  This is the volume group that
# holds the cluster lock which is used to break a cluster formation
# tie.  This volume group should not be used by any other cluster
# as cluster lock device.

FIRST_CLUSTER_LOCK_VG   /dev/vg01


# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.

NODE_NAME               ftsys9
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP        15.13.171.32
  NETWORK_INTERFACE     lan3
    HEARTBEAT_IP        192.6.7.3
  NETWORK_INTERFACE     lan4
  NETWORK_INTERFACE     lan1
    HEARTBEAT_IP        192.6.143.10
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c1t2d0


# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0


# Primary Network Interfaces on Bridged Net 1: lan0.
#   Warning: There are no standby network interfaces on bridged net 1.
# Primary Network Interfaces on Bridged Net 2: lan3.
#   Possible standby Network Interfaces on Bridged Net 2: lan4.
# Primary Network Interfaces on Bridged Net 3: lan1.
```

```
#   Warning: There are no standby network interfaces on bridged net 3.

NODE_NAME               ftsys10
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP        15.13.171.30
  NETWORK_INTERFACE     lan3
    HEARTBEAT_IP        192.6.7.4
  NETWORK_INTERFACE     lan4
  NETWORK_INTERFACE lan1
    HEARTBEAT_IP        192.6.143.20
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c1t2d0


# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0


# Primary Network Interfaces on Bridged Net 1: lan0.
#   Warning: There are no standby network interfaces on bridged net 1.
# Primary Network Interfaces on Bridged Net 2: lan3.
#   Possible standby Network Interfaces on Bridged Net 2: lan4.
# Primary Network Interfaces on Bridged Net 3: lan1.
#   Warning: There are no standby network interfaces on bridged net 3.


# Cluster Timing Parameters (microseconds).

HEARTBEAT_INTERVAL       1000000
NODE_TIMEOUT             2000000


# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT       600000000
NETWORK_POLLING_INTERVAL 2000000

# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the cluster.
# You can not add packages beyond this limit.
# This parameter is required.
MAX_CONFIGURED_PACKAGES            10


#
# List of cluster aware Volume Groups. These volume groups
# will be used by clustered applications via the vgchange -a e command.
# For example:
# VOLUME_GROUP          /dev/vgdatabase
# VOLUME_GROUP          /dev/vg02
VOLUME_GROUP            /dev/vg01
VOLUME_GROUP            /dev/vg02
```

The man page for the `cmquerycl` command lists the definitions of all the parameters that appear in this file. Many are also described in the "Planning" chapter. Modify your */etc/cmcluster/cmclconf.ascii* file to your requirements, using the data on the cluster configuration worksheet.

In the file, keywords are separated from definitions by white space. Comments are permitted, and must be preceded by a pound sign (#) in the far left column. See the man page for the `cmquerycl` command for more details.

Identifying Cluster Volume Groups

The file will include an entry for all volume groups that are to be defined as cluster-aware, that is, those which can be accessed by different nodes in the cluster at different times. A separate `VOLUME_GROUP` line should appear for each volume group that will be activated by any package running in the cluster. To leave a volume group unmarked, remove the volume group name from the ASCII file.

NOTE If a volume group is not cluster-aware, then it cannot be activated by a package control script.

Redeploying Previously Configured Volume Groups

In configuring a new cluster, if you are using volume groups that were used in a previous cluster configuration, you should ensure that they are not currently cluster aware (marked with a cluster id). You can use the following command to remove the cluster id if necessary:

```
# vgchange -c n
```

Identifying the Cluster Lock Volume Group and Disk

A cluster lock disk is required for two node clusters like the one in this example. The disk must be accessible to all nodes and must be powered separately from the nodes. Refer to the section "Use of the Cluster Lock" in Chapter 3 for additional information.

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

If necessary, you can configure a second cluster lock. Enter the following parameters in the cluster configuration file:

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

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

To display the failover times of disks, use the `cmquerycl` command, specifying all the nodes in the cluster:

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

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

Specifying Maximum Number of Configured Packages

MC/ServiceGuard preallocates memory and threads at cluster startup time. It calculates these values based on the number of packages specified in the `MAX_CONFIGURED_PACKAGES` parameter in the cluster configuration file. This value must be equal to or greater than the number of packages currently configured in the cluster. The default is 0, which means that you must enter a value if you wish to use packages. The absolute maximum number of packages per cluster is 30.

Identifying Serial Heartbeat Connections

If you are using a serial (RS232) line as a heartbeat connection, use the `SERIAL_DEVICE_FILE` parameter and enter the device file name that corresponds to the serial port you are using on each node. Be sure that the serial cable is securely attached during and after configuration.

Adding or Removing Nodes While the Cluster is Running

You can reconfigure the cluster by adding or removing nodes while the cluster is up and running. The procedures are described in the chapter on "Cluster and Package Maintenance."

Verifying the Cluster Configuration

SAM automatically checks the configuration you enter and reports any errors. If you have edited an ASCII cluster configuration file, use the following command to verify the content of the file:

```
# cmcheckconf -v -C /etc/cmcluster/cmclconf.ascii
```

This command or automatic verification in SAM both check the following:

- Network addresses and connections.
- Cluster lock connectivity.
- Validity of configuration parameters for the cluster and packages.
- Uniqueness of names.
- Existence and permission of scripts specified in the command line.
- If all nodes specified are in the same heartbeat subnet.
- If you specify the wrong configuration filename.
- If all nodes can be accessed.
- No more than one CLUSTER_NAME, HEARTBEAT_INTERVAL, and AUTO_START_TIMEOUT are specified.
- The value for HEARTBEAT_INTERVAL is at least one second.
- The value for NODE_TIMEOUT is at least twice the value of HEARTBEAT_INTERVAL.
- The value for AUTO_START_TIMEOUT variables is >=0.
- Heartbeat network minimum requirement. The cluster must have one heartbeat LAN configured with a standby, or two heartbeat LANs, or one heartbeat LAN and an RS232 connection.
- At least one NODE_NAME is specified.
- Each node is connected to each heartbeat network.
- All heartbeat networks are of the same type of LAN.
- The network interface device files specified are valid LAN device files.
- RS-232 is configured on a two node cluster, and there is no more than one serial (RS232) port connection per node.

- `VOLUME_GROUP` entries are not currently marked as cluster-aware.

If the cluster is online, SAM (or the `cmcheckconf` command) also verifies that all the conditions for the specific change in configuration have been met.

Distributing the Binary Configuration File

After specifying all cluster parameters, you use SAM or HP-UX commands to apply the configuration. This action distributes the binary configuration file to all the nodes in the cluster. We recommend doing this separately *before* you configure packages (described in the next chapter). In this way, you can verify the cluster lock, heartbeat networks, and other cluster-level operations by using the `cmviewcl` command on the running cluster. Before distributing the configuration, ensure that your security files permit copying among the cluster nodes. See "Preparing Your Systems" at the beginning of this chapter.

Distributing the Configuration File with SAM

When you have finished entering parameters in the Cluster Configuration subarea in SAM, you are asked to verify the copying of the files to all the nodes in the cluster. When you respond OK to the verification prompt, MC/ServiceGuard copies the binary configuration file and the ASCII configuration file to all the nodes in the cluster.

Distributing the Configuration File with HP-UX Commands

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 across the nodes.

```
# cmapplyconf -v -C /etc/cmcluster/cmclconf.ascii -P \
 /etc/cmcluster/pkg1/pkg1conf.ascii
```

- Deactivate the cluster lock volume group.

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

The `cmapplyconf` command creates a binary version of the cluster configuration file and distributes it to all nodes in the cluster. This action ensures that the contents of the file are consistent across all nodes. Note that the **cmapplyconf** command does not distribute the ASCII configuration file.

CAUTION

The cluster lock volume group must be activated *only* on the node from which you issue the `cmapplyconf` command, so that the lock disk can be initialized. If you attempt to configure a cluster either using SAM or by issuing the `cmapplyconf` command on one node while the lock volume group is active on another, different node, the cluster lock will be left in an unknown state. Therefore, you must ensure that when you configure the cluster, the cluster lock volume group is active only on the configuration node and deactivated on all other nodes.

Be sure to deactivate the cluster lock volume group on the configuration node after `cmapplyconf` is executed.

Storing Volume Group and Cluster Lock Configuration Data

After configuring the cluster, create a backup copy of the volume group configuration by using the `vgcfgbackup` command for each volume group you have created. If a disk in a volume group must be replaced, you can then restore the disk's metadata by using the `vgcfgrestore` command. The procedure is described under "Replacing Disks" in the "Troubleshooting" chapter.

Be sure to use `vgcfgbackup` for all volume groups, including the cluster lock volume group.

NOTE

You *must* use the `vgcfgbackup` command to store a copy of the cluster lock disk's configuration data whether you created the volume group using SAM or using HP-UX commands.

The lock disk is normally configured with redundant copies either through mirroring or RAID. 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.

Checking Cluster Operation

MC/ServiceGuard also provides several commands for manual control of the cluster:

- `cmrunnode` is used to start a node.
- `cmhaltnode` is used to manually stop a running node. (This command is also used by `shutdown(1m)`.)

- `cmruncl` is used to manually start a stopped cluster.
- `cmhaltcl` is used to manually stop a cluster.

You can use these commands to test cluster operation, as in the following:

1. If the cluster is not already online, run the cluster, as follows:

   ```
   # cmruncl -f -v
   ```

2. When the cluster has started, use the following command to ensure that cluster components are operating correctly:

   ```
   # cmviewcl -v
   ```

   Make sure that all nodes and networks are functioning as expected. For information about using `cmviewcl`, refer to the chapter on "Cluster and Package Maintenance."

3. Use the following sequence of commands to verify that nodes leave and enter the cluster as expected:

   - On a cluster node, issue the `cmhaltnode` command.
   - Use the `cmviewcl` command to verify that the node has left the cluster..
   - Issue the `cmrunnode` command.
   - Use the `cmviewcl` command again to verify that the node has returned to operation.

4. Use the following command to bring down the cluster:

   ```
   # cmhaltcl -v -f
   ```

Additional cluster testing is described in the "Troubleshooting" chapter. Refer to Appendix A for a complete list of MC/ServiceGuard commands

Preventing Automatic Activation of Volume Groups

It is important to prvent package volume groups from being activated at system boot time by the /etc/lvmrc file. To ensure that this does not happen, edit the /etc/lvmrc file on all nodes. Set `AUTO_VG_ACTIVATE` to 0, then include all the volume groups that are not cluster bound in the `custom_vg_activation` function. Volume groups that will be used by packages should *not* be included anywhere in the file, since they will be activated and deactivated by control scripts.

NOTE The root volume group does not need to be included in the `custom_vg_activation` function, since it is automatically activated before the /etc/lvmrc file is used at boot time.

Setting up Autostart Features

Automatic startup is the process in which each node individually joins a cluster. If a cluster already exists, the node attempts to join it; if no cluster is running, the node attempts to form a cluster consisting of all configured nodes. Automatic cluster start is the preferred way to start a cluster. No action is required by the system administrator. To enable automatic cluster start, set the flag `AUTOSTART_CMCLD` to 1 in the `/etc/rc.config.d/cmcluster` file; the node will then join the cluster at boot time.

In order to automate the startup of cluster nodes after a system boot, you must modify the */etc/rc.config.d/cmcluster* file on each node. MC/ServiceGuard provides this startup script to control the startup process:

```
#*************************  CMCLUSTER  ************************
# Highly Available Cluster configuration
#
# @(#) $Revision: 72.2 $
#
# AUTOSTART_CMCLD:      If set to 1, the node will attempt to
#                       join it's CM cluster automatically when
#                       the system boots.
#                       If set to 0, the node will not attempt
#                       to join it's CM cluster.
#
AUTOSTART_CMCLD=1
```

Changing the System Message

You may find it useful to modify the system's login message to include a statement such as the following:

```
This system is a node in a high availability cluster.
Halting this system may cause applications and services to
start up on another node in the cluster.
```

You might wish to include a list of all cluster nodes in this message, together with additional cluster-specific information.

The /etc/issue and /etc/motd files may be customized to include cluster-related information.

Deleting the Cluster Configuration

You can delete a cluster configuration from all cluster nodes by using SAM or by issuing the `cmdeleteconf` command. The command prompts for a verification before deleting the files unless you use the `-f` option. You can only delete the configuration when the cluster is down. The action removes the binary configuration file from all the nodes in the cluster and resets all cluster-aware volume groups to be no longer cluster-aware.

NOTE

Although the cluster must be halted, all nodes in the cluster should be powered up and accessible before you use the `cmdeleteconf` command. If a node is powered down, power it up and boot. If a node is inaccessible, you will see a list of inaccessible nodes together with the following message:

It is recommended that you do not proceed with the configuration operation unless you are sure these nodes are permanently unavailable.

Do you want to continue?

Reply Yes to remove the configuration. Later, if the inaccessible node becomes available, you should run the `cmdeleteconf` command on that node to remove the configuration file.

6 Configuring Packages and Their Services

In addition to configuring the cluster, you must identify your highly available application services, and group them into packages. This chapter describes the following *package configuration* tasks:

- Creating the Package Configuration
- Writing the Package Control Script
- Distributing the Binary Cluster Configuration File

Each of these tasks is described in a separate section below.

In configuring your own packages, use data from the Package Configuration Worksheet described in the "Planning" chapter. Package configuration data from the worksheet becomes part of the binary cluster configuration file on all nodes in the cluster. The control script data from the worksheet goes into an executable package control script which runs specific applications (services) and monitors their operation.

Creating the Package Configuration

The package configuration process defines a set of application services that are run by the package manager when a package starts up on a node in the cluster. The configuration also includes a prioritized list of cluster nodes on which the package can run together with definitions of the acceptable types of failover allowed for the package.

You can create a package using SAM or using HP-UX commands and editors. The following section describes SAM configuration. If you are using MC/ServiceGuard commands, skip ahead to the section entitled "Using MC/ServiceGuard Commands to Create a Package."

Using SAM to Configure a Package

To configure a high availability package use the following steps on the configuration node (ftsys9):

1. In SAM, choose the "Clusters" area, then the High Availability Clusters option.
2. Choose the Package Configuration option. SAM displays a Package Configuration screen. If no packages have yet been configured, the list area will be empty. If there are one or more MC/ServiceGuard packages already configured on clusters in your network, you will see them listed.
3. Select the Actions menu, and choose Create/Add a Package. A step menu appears.
4. Choose each required step in sequence, filling in the dialog boxes with required information, or accepting the default values shown. For information about each step, choose Help.
5. When finished with all steps, select **OK** at the Step Menu screen. This action creates (or modifies) the package configuration file and then copies the file to all the nodes in the cluster. This action also creates a package control script which is copied to all nodes.
6. When the file copying is finished, you return to the Package Configuration screen.
7. Exit from the Package Configuration screen, returning to the High Availability Clusters menu.

Skip ahead to the section on "Writing the Package Control Script."

Using MC/ServiceGuard Commands to Create a Package

Use the following procedure to create packages by editing and processing a package configuration file.

1. First, create a subdirectory for each package you are configuring in the /etc/cmcluster directory:

   ```
   # mkdir /etc/cmcluster/pkg1
   ```

   You can use any directory names you wish.

2. Next, generate a package configuration template for the package:

   ```
   # cmmakepkg -p /etc/cmcluster/pkg1/pkg1conf.ascii
   ```

   You can use any file names you wish for the ASCII templates.

3. Edit these template files to specify package name, prioritized list of nodes, the location of the control script, and failover parameters for each package. Include the data recorded on the Package Configuration Worksheet.

Configuring in Stages

It is recommended to configure packages on the cluster in stages, as follows:

1. Configure volume groups and mount points only.
2. Apply the configuration.
3. Distribute the control script to all nodes.
4. Run the cluster and ensure that packages can be moved from node to node.
5. Halt the cluster.
6. Configure package IP addresses and application services in the control script.
7. Distribute the control script to all nodes.
8. Run the cluster and ensure that applications run as expected and that packages fail over correctly when services are disrupted.

Package Configuration Template File

The following is a sample package configuration file template customized for a typical package.

```
# **********************************************************************
# ****** HIGH AVAILABILITY PACKAGE CONFIGURATION FILE (template) *******
# **********************************************************************
# ******* Note: This file MUST be edited before it can be used. ********
# * For complete details about package parameters and how to set them, *
# * consult the MC/ServiceGuard or MC/LockManager manpages or manuals. *
# **********************************************************************

# Enter a name for this package.  This name will be used to identify the
# package when viewing or manipulating it.  It must be different from
# the other configured package names.

PACKAGE_NAME  pkg1


# Enter the names of the nodes configured for this package.  Repeat
# this line as necessary for additional adoptive nodes.
# Order IS relevant.  Put the second Adoptive Node AFTER the first
# one.
# Example : NODE_NAME  original_node
#           NODE_NAME  adoptive_node

NODE_NAME ftsys9
NODE_NAME ftsys10


# Enter the complete path for the run and halt scripts.  In most cases
# the run script and halt script specified here will be the same script,
# the package control script generated by the cmmakepkg command.  This
# control script handles the run(ning) and halt(ing) of the package.
# If the script has not completed by the specified timeout value,
# it will be terminated.  The default for each script timeout is
# NO_TIMEOUT.  Adjust the timeouts as necessary to permit full
# execution of each script.
# Note: The HALT_SCRIPT_TIMEOUT should be greater than the sum of
# all SERVICE_HALT_TIMEOUT specified for all services.

RUN_SCRIPT   /etc/cmcluster/pkg1/control.sh
RUN_SCRIPT_TIMEOUT      NO_TIMEOUT
HALT_SCRIPT  /etc/cmcluster/pkg1/control.sh
HALT_SCRIPT_TIMEOUT     NO_TIMEOUT


# Enter the SERVICE_NAME, the SERVICE_FAIL_FAST_ENABLED and the
# SERVICE_HALT_TIMEOUT values for this package.  Repeat these
# three lines as necessary for additional service names.  All
# service names MUST correspond to the service names used by
# cmrunserv and cmhaltserv commands in the run and halt scripts.
#
# The value for SERVICE_FAIL_FAST_ENABLED can be either YES or
# NO.  If set to YES, in the event of a service failure, the
# cluster software will halt the node on which the service is
# running.  If SERVICE_FAIL_FAST_ENABLED is not specified, the
# default will be NO.
#
# SERVICE_HALT_TIMEOUT is represented in the number of seconds.
# This timeout is used to determine the length of time (in
# seconds) the cluster software will wait for the service to
# halt before a SIGKILL signal is sent to force the termination
# of the service.  In the event of a service halt, the cluster
# software will first send a SIGTERM signal to terminate the
# service.  If the service does not halt, after waiting for the
# specified SERVICE_HALT_TIMEOUT, the cluster software will send
# out the SIGKILL signal to the service to force its termination.
# This timeout value should be large enough to allow all cleanup
```

```
# processes associated with the service to complete.  If the
# SERVICE_HALT_TIMEOUT is not specified, a zero timeout will be
# assumed, meaning the cluster software will not wait at all
# before sending the SIGKILL signal to halt the service.
#
# Example: SERVICE_NAME                   DB_SERVICE
#          SERVICE_FAIL_FAST_ENABLED      NO
#          SERVICE_HALT_TIMEOUT           300
#
# To configure a service, uncomment the following lines and
# fill in the values for all of the keywords.
#
SERVICE_NAME                    service1
SERVICE_FAIL_FAST_ENABLED       NO
SERVICE_HALT_TIMEOUT            300


# Enter the network subnet name that is to be monitored for this package.
# Repeat this line as necessary for additional subnet names.  If any of
# the subnets defined goes down, the package will be switched to another
# node that is configured for this package and has all the defined subnets
# available.

#SUBNET   15.16.168.0


# The following keywords (RESOURCE_NAME, RESOURCE_POLLING_INTERVAL, and
# RESOURCE_UP_VALUE) are used to specify Package Resource Dependencies.  To
# define a Package Resource Dependency, a RESOURCE_NAME line with a fully
# qualified resource path name, and one or more RESOURCE_UP_VALUE lines are
# required.  A RESOURCE_POLLING_INTERVAL line (how often in seconds the resource
# is to be monitored) is optional and defaults to 60 seconds.  An operator and
# a value are used with RESOURCE_UP_VALUE to define when the resource is to be
# considered up.  The operators are =, !=, >, <, >=, and <=, depending on the
# type of value.  Values can be string or numeric.  If the type is string, then
# only = and != are valid operators.  If the string contains whitespace, it
# must be enclosed in quotes.  String values are case sensitive.  For example,
#
# Resource is up when its value is
# --------------------------------
# RESOURCE_UP_VALUE= UP"UP"
# RESOURCE_UP_VALUE!= DOWNAny value except "DOWN"
# RESOURCE_UP_VALUE= "On Course""On Course"
#
# If the type is numeric, then it can specify a threshold, or a range to
# define a resource up condition.  If it is a threshold, then any operator
# may be used.  If a range is to be specified, then only > or >= may be used
# for the first operator, and only < or <= may be used for the second operator.
# For example,
# Resource is up when its value is
# --------------------------------
# RESOURCE_UP_VALUE     = 55    (threshold)
# RESOURCE_UP_VALUE     > 5.1greater than 5.1    (threshold)
# RESOURCE_UP_VALUE     > -5 and < 10between -5 and 10   (range)
#
# Note that "and" is required between the lower limit and upper limit
# when specifying a range.  The upper limit must be greater than the lower
# limit.  If RESOURCE_UP_VALUE is repeated within a RESOURCE_NAME block, then
# they are inclusively OR'd together.  Package Resource Dependencies may be
# defined by repeating the entire RESOURCE_NAME block.
#
# Example : RESOURCE_NAME/net/lan/lan0/res1
#     RESOURCE_POLLING_INTERVAL120
#     RESOURCE_UP_VALUE= RUNNING
#     RESOURCE_UP_VALUE= ONLINE
#
#     Means that the value of resource /net/lan/lan0/res1 will be
#    checked every 120 seconds, and is considered to be 'up' when
#    its value is "RUNNING" or "ONLINE".
#
# Uncomment the following lines to specify Package Resource Dependencies.
#
```

```
#RESOURCE_NAME     <Full_path_name>
#RESOURCE_POLLING_INTERVAL  <numeric_seconds>
#RESOURCE_UP_VALUE     <op> <string_or_numeric> [and <op> <numeric>]


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

PKG_SWITCHING_ENABLED   YES


# The default for NET_SWITCHING_ENABLED is YES.  In the event of a
# failure, this permits the cluster software to switch LANs locally
# (transfer to a standby LAN card).  Adjust as necessary.

NET_SWITCHING_ENABLED   YES


# The default for NODE_FAIL_FAST_ENABLED is NO.  If set to YES,
# in the event of a failure, the cluster software will halt the node
# on which the package is running.  Adjust as necessary.

NODE_FAIL_FAST_ENABLED   NO
```

Use the information on the Package Configuration worksheet to complete the file. Refer also to the comments on the configuration template for additional explanation of each parameter. You may include the following information:

- NODE_NAME. Enter the name of each node in the cluster on a separate line.
- RUN_SCRIPT and HALT_SCRIPT. Specify the pathname of the package control script (described in the next section). No default is provided.
- If your package contains services, enter the SERVICE_NAME, SERVICE_FAIL_FAST_ENABLED and SERVICE_HALT_TIMEOUT. values. Enter a group of these three for each service. You can configure no more than 30 services per package.
- If your package has IP addresses associated wtih it, enter the SUBNET. This must be a subnet that is already specified in the cluster confiugration.
- NODE_FAIL_FAST_ENABLED parameter. Enter YES or NO.

To configure monitoring within the package for a registered resource, enter values for the following parameters. Use the "Additional Package Resources" part of the "Package Configuration" subarea in SAM to obtain a list of resource names, polling intervals, and up values that you can set up as dependencies for your packages.

- RESOURCE_NAME. Enter the name of a registered resource that is to be monitored by MC/ServiceGuard.
- RESOURCE_POLLING_INTERVAL. Enter the time between attempts to assure that the resource is healthy.

- RESOURCE_UP_VALUE. Enter the value or values that determine when the resource is considered to be up. During monitoring, if a different value is found for the resource, the package will fail.

This package configuration data is later combined with the cluster configuration data in the binary cluster configuration file.

Writing the Package Control Script

The package control script contains all the information necessary to run all the services in the package, monitor them during operation, react to a failure, and halt the package when necessary. You can use either SAM or HP-UX commands to create or modify the package control script. For security reasons, the control script must reside in a directory with the string *cmcluster* in the path.

Using SAM to Write the Package Control Script

Select the High Availability options in SAM, then choose "Package Configuration." From the Action menu, choose "Create/Add a Package." The step menu appears, showing a group of options. The last two steps on the menu are for choosing the pathname of the package control script. Select each option after you define the package itself. For more information, use the Help key.

When you create a package control script this way, you do not need to do any further editing, but you may customize the script if you wish.

Using Commands to Write the Package Control Script

Each package must have a separate control script. The control script is placed in the package directory and is given the same name that it has in the package configuration file. The package control script contains both the run instructions and the halt instructions for the package. It must be executable. Use the following procedure to create a control scripts for the sample package *pkg1*.

First, generate a control script template:

```
# cmmakepkg -s /etc/cmcluster/pkg1/control.sh
```

You may customize the script, as described in the next section.

Customizing the Package Control Script

Check the definitions and declarations at the beginning of the control script using the information in the Package Configuration worksheet. You need to customize as follows:

- Update the PATH statement to reflect any required paths needed to start your services.
- Enter the names of volume groups that will be activated.

- Add the names of logical volumes and file systems that will be mounted on them.
- Define IP subnet and IP address pairs for your package.
- Add service name(s).
- Add service command(s)
- Add a service restart parameter, if desired.

NOTE

Use care in defining service run commands. Each run command is executed by the control script in the following way:

- The `cmrunserv` command executes each run command and then monitors the process id of the process created by the run command.
- When the command started by `cmrunserv` exits, MC/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, MC/ServiceGuard will consider this normal exit as a *failure*.

To avoid problems in the execution of control scripts, ensure that each run command is the name of an actual service and that its process remains alive until the actual service stops.

If you need to define a set of run and halt operations in addition to the defaults, create functions for them in the sections under the heading `CUSTOMER DEFINED FUNCTIONS`. The next section shown an excerpt from the control script for a sample package configuration.

Package Control Script Template File

```
#"(#) A.10.10         SRevision: 80.8 $ SDate: 97/07/17 08:45:04 $"
# ************************************************************************
# *                                                                      *
# *        HIGH AVAILABILITY PACKAGE CONTROL SCRIPT (template)           *
# *                                                                      *
# *       Note: This file MUST be edited before it can be used.          *
# *                                                                      *
# ************************************************************************

# UNCOMMENT the variables as you set them.

# Set PATH to reference the appropriate directories.
PATH=/sbin:/usr/bin:/usr/sbin:/etc:/bin

# VOLUME GROUP ACTIVATION:
# Specify the method of activation for volume groups.
# Leave the default ("VGCHANGE="vgchange -a e") if you want volume
# groups activated in exclusive mode. This assumes the volume groups have
# been initialized with 'vgchange -c y' at the time of creation.
#
```

```
# Uncomment the first line (VGCHANGE="vgchange -a e -q n"), and comment
# out the default, if your disks are mirrored on separate physical paths,
#
# Uncomment the second line (VGCHANGE="vgchange -a y") if you wish to
# use non-exclusive activation mode. Single node cluster configurations
# must use non-exclusive activation.
#
# VGCHANGE="vgchange -a e -q n"
# VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"# Default

# VOLUME GROUPS
# Specify which volume groups are used by this package. Uncomment VG[0]=""
# and fill in the name of your first volume group. You must begin with
# VG[0], and increment the list in sequence.
#
# For example, if this package uses your volume groups vg01 and vg02, enter:
#         VG[0]=vg01
#         VG[1]=vg02
#
# The volume group activation method is defined above. The filesystems
# associated with these volume groups are specified below.
#
#VG[0]=""

# FILESYSTEMS
# Specify the filesystems which are used by this package. Uncomment
# LV[0]=""; FS[0]=""; FS_MOUNT_OPT[0]="" and fill in the name of your first
# logical volume, filesystem and mount option for the file system. You must
# begin with LV[0], FS[0] and FS_MOUNT_OPT[0] and increment the list in
# sequence.
#
# For example, if this package uses the file systems pkg1a and pkg1b,
# which are mounted on the logical volumes lvol1 and lvol2 with read and
# write options enter:
#          LV[0]=/dev/vg01/lvol1; FS[0]=/pkg1a; FS_MOUNT_OPT[0]="-o rw"
#          LV[1]=/dev/vg01/lvol2; FS[1]=/pkg1b; FS_MOUNT_OPT[1]="-o rw"
#
# The filesystems are defined as triplets of entries specifying the logical
# volume, the mount point and the mount options for the file system. Each
# filesystem will be fsck'd prior to being mounted. The filesystems will be
# mounted in the order specified during package startup and will be unmounted
# in reverse order during package shutdown. Ensure that volume groups
# referenced by the logical volume definitions below are included in
# volume group definitions above.
#
#LV[0]=""; FS[0]=""; FS_MOUNT_OPT[0]=""

# IP ADDRESSES
# Specify the IP and Subnet address pairs which are used by this package.
# Uncomment IP[0]="" and SUBNET[0]="" and fill in the name of your first
# IP and subnet address. You must begin with IP[0] and SUBNET[0] and
# increment the list in sequence.
#
# For example, if this package uses an IP of 192.10.25.12 and a subnet of
# 192.10.25.0 enter:
#          IP[0]=192.10.25.12
#          SUBNET[0]=192.10.25.0 # (netmask=255.255.255.0)
#
# Hint: Run "netstat -i" to see the available subnets in the Network field.
#
# IP/Subnet address pairs for each IP address you want to add to a subnet
# interface card.  Must be set in pairs, even for IP addresses on the same
# subnet.
#
#IP[0]=""
#SUBNET[0]=""

# SERVICE NAMES AND COMMANDS.
# Specify the service name, command, and restart parameters which are
# used by this package. Uncomment SERVICE_NAME[0]="", SERVICE_CMD[0]="",
# SERVICE_RESTART[0]="" and fill in the name of the first service, command,
```

```
# and restart parameters. You must begin with SERVICE_NAME[0], SERVICE_CMD[0],
# and SERVICE_RESTART[0] and increment the list in sequence.
#
# For example:
#          SERVICE_NAME[0]=pkg1a
#          SERVICE_CMD[0]="/usr/bin/X11/xclock -display 192.10.25.54:0"
#          SERVICE_RESTART[0]=""  # Will not restart the service.
#
#          SERVICE_NAME[1]=pkg1b
#          SERVICE_CMD[1]="/usr/bin/X11/xload -display 192.10.25.54:0"
#          SERVICE_RESTART[1]="-r 2"   # Will restart the service twice.
#
#          SERVICE_NAME[2]=pkg1c
#          SERVICE_CMD[2]="/usr/sbin/ping"
#          SERVICE_RESTART[2]="-R" # Will restart the service an infinite
#                                  number of times.
#
# Note: No environmental variables will be passed to the command, this
# includes the PATH variable. Absolute path names are required for the
# service command definition.  Default shell is /usr/bin/sh.
#
#SERVICE_NAME[0]=""
#SERVICE_CMD[0]=""
#SERVICE_RESTART[0]=""

# DTC manager information for each DTC.
# Example: DTC[0]=dtc_20
#DTC_NAME[0]=
```

The above excerpt from the control script shows the assignment of values to a set of variables. The remainder of the script uses these variables to control the package by executing Logical Volume Manager commands, HP-UX commands, and MC/ServiceGuard commands including `cmrunserv`, `cmmodnet`, and `cmhaltserv`. Examine a copy of the control script template to see the flow of logic. Use the following command:

```
# cmmakepkg -s | more
```

The main function appears at the end of the script.

Note that individual variables are optional; you should include only as many as you need for proper package operation. For example, if your package does not need to activate a volume group, omit the `VG` variables; if the package does not use services, omit the corresponding `SERVICE_NAME`, `SERVICE_CMD`, and `SERVICE_RESTART` variables; and so on.

After customizing the script, distribute it to each node in the cluster using `rcp`, `ftp`, or your favorite method of copying.

Adding Customer Defined Functions to the Package Control Script

You can add additional shell commands to the package control script to be executed whenever the package starts or stops. Simply enter these commands in the `CUSTOMER DEFINED FUNCTIONS` area of the script. This gives you the ability to further customize the control script.

An example of this portion of the script is shown below, with the `date` and `echo` commands included to log starts and halts of the package to a special file.

```
# START OF CUSTOMER DEFINED FUNCTIONS

# This function is a place holder for customer defined functions.
# You should define all actions you want to happen here, before the service is
# started.  You can create as many functions as you need.

function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.
 date >> /tmp/pkg1.datelog
 echo 'Starting pkg1' >> /tmp/pkg1.datelog
 test_return 51
}

# This function is a place holder for customer defined functions.
# You should define all actions you want to happen here, before the service is
# halted.

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.
: # do nothing instruction, because a function must contain some command.
 date >> /tmp/pkg1.datelog
 echo 'Halting pkg1' >> /tmp/pkg1.datelog
 test_return 52
}

# END OF CUSTOMER DEFINED FUNCTIONS
```

Adding MC/ServiceGuard Commands in Customer Defined Functions

You can add MC/ServiceGuard commands (such as `cmmodpkg`) in the Customer Defined Functions section of a package control script. However, these commands must not interact with the package itself. Additionally, if an MC/ServiceGuard command interacts with another package, then you need to check all packages with MC/ServiceGuard commands for the possibility of a command loop.

For instance, a command loop might occur under the following circumstances. Suppose Pkg1 does a `cmmodpkg -d` of Pkg2, and Pkg2 does a `cmmodpkg -d` of Pkg1. If both Pkg1 and Pkg2 start at the same time, Pkg1 tries to `cmmodpkg` Pkg2. However, that `cmmodpkg` command has to wait for Pkg2 startup to complete. Pkg2 tries to `cmmodpkg` Pkg1, but Pkg2 has to wait for Pkg1 startup to complete, thereby causing a command loop.

To avoid this situation, it is a good idea to alway specify a RUN_SCRIPT_TIMEOUT and a HALT_SCRIPT_TIMEOUT for all packages, especially packages that use ServiceGuard commands in their control scripts. If a timeout is not specified and your configuration has a command loop as described above, inconsistent results can occur, including a hung cluster.

Adding or Removing Packages on a Running Cluster

You can add or remove packages while the cluster is running, subject to the limit of MAX_CONFIGURED_PACKAGES. To add or remove packages online, refer to the chapter on "Cluster and Package Maintenance."

Verify and Distribute the Configuration

You can use SAM or HP-UX commands to verify and distribute the binary cluster configuration file among the nodes of the cluster.

Distributing the Configuration File And Control Script with SAM

When you have finished creating a package in the Package Configuration subarea in SAM, you are asked to verify the copying of the files to all the nodes in the cluster. When you respond OK to the verification prompt, MC/ServiceGuard copies the binary configuration file and package control script to all the nodes in the cluster.

Copying Package Control Scripts with HP-UX commands

Use HP-UX commands to copy package control scripts from the configuration node to the same pathname on all nodes which can possibly run the package. Use your favorite method of file transfer (e. g., `rcp` or `ftp`). For example, from *ftsys9*, you can issue the `rcp` command to copy the package control script to *ftsys10*:

```
# rcp /etc/cmcluster/pkg1/control.sh \
ftsys10:/etc/cmcluster/pkg1/control.sh
```

Distributing the Binary Cluster Configuration File with HP-UX Commands

Use the following steps from the node on which the ASCII cluster and package configuration files exist:

- Verify that the configuration file is correct. Use the following command:

  ```
  # cmcheckconf -C /etc/cmcluster/cmclconf.ascii -P \
   /etc/cmcluster/pkg1/pkg1conf.ascii
  ```

- Activate the cluster lock volume group so that the lock disk can be initialized:

  ```
  # vgchange -a y /dev/vg01
  ```

- Generate the binary configuration file and distribute it across the nodes.

  ```
  # cmapplyconf -v -C /etc/cmcluster/cmclconf.ascii -P \
   /etc/cmcluster/pkg1/pkg1conf.ascii
  ```

- Deactivate the cluster lock volume group.

```
# vgchange -a n /dev/vg01
```

The `cmapplyconf` command creates a binary version of the cluster configuration file and distributes it to all nodes in the cluster. This action ensures that the contents of the file are consistent across all nodes.

NOTE `cmcheckconf` and `cmapplyconf` must be used again any time changes are made to the cluster and package configuration files.

Verifying Cluster and Package Configuration

While configuring your MC/ServiceGuard cluster, it's a good idea to test that the various components of the cluster behave correctly in case of a failure. See the chapter "Troubleshooting Your Cluster" for an explanation of how to test that your cluster responds properly in the event of a package failure, a node failure, or a LAN failure.

7 Cluster and Package Maintenance

This chapter describes how to 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:

- Managing the Cluster and Nodes
- Reconfiguring a Running Cluster
- Managing Packages and Services
- Reconfiguring Packages in a Running Cluster
- Responding to Cluster Events
- Single Node Operation
- Removing MC/ServiceGuard Software from a System

All administration tasks can be carried out using SAM or HP-UX commands. From the SAM main menu, choose Clusters, then High Availability Clusters, and select the appropriate type of administration — cluster or package administration.

Managing the Cluster and Nodes

Managing the cluster involves the following tasks:

- Starting the Cluster When All Nodes are Down
- Adding Previously Configured Nodes to a Running Cluster
- Removing Nodes from Operation in a Running Cluster
- Halting the Entire Cluster
- Reconfiguring a Halted Cluster

You can use SAM or MC/ServiceGuard commands to start or stop the cluster, and to modify the cluster configuration. Starting the cluster means running the cluster daemon on one or more of the nodes in a cluster. You use different MC/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.

Starting the Cluster When all Nodes are Down

You can use SAM or MC/ServiceGuard commands to start the cluster.

Using SAM to Start the Cluster

In SAM, choose Clusters, then High Availability Clusters. In the Cluster Administration area, highlight the cluster you wish to start, then select "Start Cluster" from the Actions menu. Indicate whether you wish to bring up the cluster on all nodes or on a subset of nodes. If you choose a subset, then you must select the nodes from the list. This option should be used only when you are sure that the cluster is not already running on any node.

At the verification prompt, select OK to start the cluster.

Using MC/ServiceGuard Commands to Start the Cluster

Use the `cmruncl` command to start the cluster when all cluster nodes are down. Use the `-n` option to specify a particular group of nodes. Without this option, all nodes will be started. The following starts all nodes configured in the cluster:

```
# cmruncl -v
```

The following example starts up the locally configured cluster only on *ftsys9* and *ftsys10*. (This form of the command should only be used when you are sure that the cluster is not already running on any node.)

```
# cmruncl -v -n ftsys9 -n ftsys10
```

WARNING

MC/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 SAM or MC/ServiceGuard commands to bring a configured node up within a running cluster.

Using SAM to Add Previously Configured Nodes to a Running Cluster

In the Cluster Administration area, highlight the cluster you are adding the node to, then select "Specify Nodes to Join the Cluster." Choose the node or nodes you are adding, then choose OK. Only nodes that are members of the current cluster configuration may be selected.

Using MC/ServiceGuard Commands to Add Previously Configured Nodes to a Running Cluster

Use the `cmrunnode` command to add one or more nodes to an already running cluster. Any node you add must already be a part of the cluster configuration. The following example adds node *ftsys8* to the cluster that was just started with only nodes *ftsys9* and *ftsys10*:

```
# cmrunnode -v ftsys8
```

Since the node's cluster is already running, the node joins the cluster and packages may be started. If the node does not find its cluster running, or the node is not part of the cluster configuration, the command fails.

Removing Nodes from Operation in a Running Cluster

You can use SAM or HP-UX commands to remove nodes from operation in a cluster. This operation removes the node from cluster operation by halting the cluster daemon, but it does not modify the cluster configuration. To remove a node from the cluster configuration permanently, you must recreate the cluster configuration file. See the next section.

Using SAM to Remove Nodes from Operation

In the Cluster Administration area, highlight the cluster you are interested in, then select "Specify Nodes to Leave the Cluster" from the Actions menu. Choose the node or nodes, then select OK.

If a package is running on a node you are shutting down, you will see a message asking for a verification that you wish to shut down the node anyway. Reply OK to verify shutdown. Following shutdown, any package that can switch to an adoptive node will start up on the adoptive node.

Using MC/ServiceGuard Commands to Remove Nodes from Operation

Use the `cmhaltnode` command to halt one or more nodes in a cluster. The cluster daemon on the specified node stops, and the node is removed from active participation in the cluster.

To halt a node with a running package, use the `-f` option. If a package was running that can be switched to an adoptive node, the switch takes place and the package starts on the adoptive node. For example, the following command causes the ServiceGuard daemon running on node *ftsys9* in the sample configuration to halt and the package running on *ftsys9* to move to *ftsys10*:

```
# cmhaltnode -f -v  ftsys9
```

This halts any packages running on the node *ftsys9* by executing the halt instructions in each package's control script. *ftsys9* is halted and the packages start on the adoptive node, *ftsys10*.

The use of `cmhaltnode` is a convenient way of bringing a node down for system maintenance while keeping its packages available on other nodes. After maintenance, the package can be returned to its primary node. See "Moving a Package," below.

To return a node to the cluster, use `cmrunnode`.

Halting the Entire Cluster

You can use SAM or MC/ServiceGuard commands to halt a running cluster.

Using SAM to Halt the Cluster

In the Cluster Administration area, highlight the cluster you wish to halt, then select "Shut Down Cluster" from the Actions menu to halt the selected cluster on all nodes. At the verification prompt, choose OK.

Using MC/ServiceGuard Commands to Halt a Cluster

The `cmhaltcl` command can be used to halt the entire cluster. This command causes all nodes in a configured cluster to halt their MC/ServiceGuard daemons. You can use the `-f` option to force the cluster to halt even when packages are running. This command can be issued from any running node. Example:

```
# cmhaltcl -f -v
```

This halts all nodes that are configured in the cluster.

Reconfiguring a Halted Cluster

You can also make a permanent change in cluster configuration when the cluster is halted. This procedure *must* be used for changes to the lock disk configuration, changes in timing prarmeters, and changes to the Maximum Number of Packages parameter, but it can be used for any other cluster configuration changes as well.

Use the following steps:

1. Halt the cluster on all nodes.
2. On one node, reconfigure the cluster as described in the chapter "Building an HA Cluster Configuration." You can do this by using SAM or by issuing the `cmquerycl` command to generate an ASCII file, which you then edit.
3. Make sure that all nodes listed in the cluster configuration file are powered up and accessible. Use SAM or the `cmapplyconf` command to copy the binary cluster configuration file to all nodes. This file overwrites any previous version of the binary cluster configuration file.
4. Use SAM or the `cmruncl` command to start the cluster on all nodes or on a subset of nodes, as desired.

Using SAM to Change the Maximum Packages Parameter

After halting the cluster, go to the Cluster Configuration area, and choose "Modify Cluster Configuration" from the Actions menu. Then select "Modify Cluster Package Requirements." Enter the new maximum number of packages in the space provided, and click OK.

Using MC/ServiceGuard Commands to Change MAX_CONFIGURED_PACKAGES

Use the `cmgetconf` command to obtain a current copy of the cluster's existing configuration. Example:

```
# cmgetconf -C clconfig.ascii
```

Edit the `clconfig.ascii` file to include the desired value for MAX_CONFIGURED_PACKAGES. Then use the `cmcheckconf` command to verify the new configuration. Use the `cmapplyconf` command to apply the changes to the configuration and send the new configuration file to all cluster nodes.

Automatically Restarting the Cluster

You can configure your cluster to automatically restart after an event, such as a long-term power failure, which brought down all nodes in the cluster. This is done by setting `AUTOSTART_CMCLD` to 1 in the `/etc/rc.config.d/cmcluster` file.

Reconfiguring a Running Cluster

You can add new nodes to the cluster configuration or delete nodes from the cluster configuration while the cluster is up and running. Note the following, however:

- You cannot change the lock disk configuration while the cluster is running.
- You cannot remove an active node from the cluster. You must halt the node first.
- You cannot delete an active volume group from the cluster configuration. You must halt any package that uses the volume group and ensure that the volume is inactive before deleting it.
- You cannot change cluster timing parameters.
- The only configuration change allowed while a node is unreachable (for example, completely disconnected from the network) is to delete the unreachable node from the cluster configuration. If there are also packages that depend upon that node, the package configuration must also be modified to delete the node. This all must be done in one configuration request (`cmapplyconf` command).

Changes to the package configuration are described in a later section.

Table 7-1 Types of Changes to Permanent Cluster Configuration

Change to the Cluster Configuration	Required Cluster State
Add a new node	All cluster nodes must be running. Serial heartbeat must not be configured.
Delete a node	A node can be deleted even though it is unavailable or unreachable.
Change Maximum Configured Packages	Cluster must not be running.
Change Timing Parameters	Cluster must not be running.
Change Cluster Lock Configuration	Cluster must not be running.
Change serial device files	Cluster must not be running
Change IP addresses for heartbeats or monitored subnets	Cluster must not be running

The following sections describe how to perform dynamic reconfiguration tasks using SAM or using MC/ServiceGuard commands.

Using SAM to Add Nodes to the Configuration While the Cluster is Running

Use the Cluster Configuration area of SAM, and choose "Modify Cluster Configuration" from the Actions menu. Then select "Modify Cluster Name and Nodes." Highlight the name of the node you wish to add to the cluster in the list on the right of the screen, then click the Add button. When you are finished changing the node list, click OK.

Using MC/ServiceGuard Commands to Add Nodes to the Configuration While the Cluster is Running

Use the following procedure to add a node with HP-UX commands. For this example, nodes *ftsys8* and *ftsys9* are already configured in a running cluster named *cluster1*, and you are adding node *ftsys10*.

1. Use the following command to store a current copy of the existing cluster configuration in a temporary file:

   ```
   # cmgetconf -C temp.ascii
   ```

2. Specify a new set of nodes to be configured and generate a template of the new configuration:

   ```
   # cmquerycl -C clconfig.ascii -c cluster1 \
     -n ftsys8 -n ftsys9 -n ftsys10
   ```

3. Edit the file `clconfig.ascii` to check the information about the new node.

4. Verify the new configuration:

   ```
   # cmcheckconf -C clconfig.ascii
   ```

5. Apply the changes to the configuration and send the new binary configuration file to all cluster nodes:

   ```
   # cmapplyconf -C clconfig.ascii
   ```

Use `cmrunnode` to start the new node, and, if desired, set the AUTOSTART_CMCLD parameter to 1 in the `/etc/rc.config.d/cmcluster` file to enable the new node to join the cluster automatically each time it reboots.

Using SAM to Delete Nodes from the Configuration While the Cluster is Running

In the Cluster Administration area of SAM, halt the node that you wish to remove from the cluster configuration. Then go to the Cluster Configuration area, and choose "Modify Cluster Configuration" from the Actions menu. Then select "Modify Cluster Name and Nodes." Highlight the name of the node you wish to delete from the cluster in the list of configured nodes, then click the Remove button. When you are finished changing the node list, click OK.

If the node you wish to delete is unreachable (disconnected from the LAN, for example), you can use SAM to delete the node only if there are no packages which specify the unreachable node. If there are packages that depend on the unreachable node, use MC/ServiceGuard commands as described in the next section.

Using MC/ServiceGuard Commands to Delete Nodes from the Configuration While the Cluster is Running

Use the following procedure to delete a node with HP-UX commands. For this example, nodes *ftsys8*, *ftsys9* and *ftsys10* are already configured in a running cluster named *cluster1*, and you are deleting node *ftsys10*.

1. Use the following command to store a current copy of the existing cluster configuration in a temporary file:

   ```
   # cmgetconf -C temp.ascii
   ```

2. Specify the new set of nodes to be configured (omitting *ftsys10*) and generate a template of the new configuration:

   ```
   # cmquerycl -C clconfig.ascii -c cluster1 -n ftsys8 -n ftsys9
   ```

3. Edit the file `clconfig.ascii` to check the information about the new node.

4. Verify the new configuration:

   ```
   # cmcheckconf -C clconfig.ascii
   ```

5. Apply the changes to the configuration and send the new binary configuration file to all cluster nodes:

   ```
   # cmapplyconf -C clconfig.ascii
   ```

Use `cmrunnode` to start the new node, and, if desired, set the AUTOSTART_CMCLD parameter to 1 in the `/etc/rc.config.d/cmcluster` file to enable the new node to join the cluster automatically each time it reboots.

NOTE

If you are attempting to remove an unreachable node that has many packages dependent on it, especially if the dependent packages use a large number of EMS resources, you may see the following message:

```
The configuration change is too large to process while the cluster is running.
Split the configuration change into multiple requests or halt the cluster.
```

In this situation, you must halt the cluster to remove the unreachable node.

Using SAM to Change the Volume Group Configuration While the Cluster is Running

In SAM, go to the Cluster Configuration area, and choose "Modify Cluster Configuration" from the Actions menu. Then select "Modify Cluster Package Requirements." You can add or delete volume groups by highlighting the volume group name and adding it to or removing it from the list of cluster-aware volume groups.

You *cannot* change the cluster lock volume group or physical volume configuration while the cluster is running.

NOTE

If you are removing a volume group from the cluster configuration, make sure that you also modify or delete any package control script that activates and deactivates this volume group. In addition, you should use the LVM `vgexport` command on the removed volume group from each node that will no longer be using the volume group.

Using MC/ServiceGuard Commands to Change the Volume Group Configuration While the Cluster is Running

Use the `cmgetconf` command to obtain a current copy of the cluster's existing configuration. Example:

```
# cmgetconf -C clconfig.ascii
```

Edit the file `clconfig.ascii` to add or delete volume groups. Then use the `cmcheckconf` command to verify the new configuration. Use the `cmapplyconf` command to apply the changes to the configuration and send the new configuration file to all cluster nodes.

NOTE If you are deleting from the cluster a volume group that is currently activated by a package, the configuration will be changed but the deletion will not take effect until the package is halted; thereafter, the package will no longer be able to run without further modification, such as removing the volume group from the package control script.

Managing Packages and Services

Managing packages and services involves the following tasks:

- Starting a Package
- Halting a Package
- Moving a Package
- Adding a Package to a Running Cluster
- Deleting a Package from a Running Cluster
- Reconfiguring a Package on a Running Cluster
- Reconfiguring a Package on a Halted Cluster

You can use SAM or MC/ServiceGuard commands to perform these tasks.

Starting a Package

Ordinarily, a package configured as part of the cluster will start up on its primary node when the cluster starts up. You may need to start a package manually after it has been halted manually. You can do this either in SAM or with MC/ServiceGuard commands.

Using SAM to Start a Package

In SAM, select "Package Administration," then choose the package you wish to start. From the "Actions" menu, choose "Start Package." If you wish to start the package on a specific node, choose "Start a Package on a Specific Node." Otherwise, choose "Start Package," and reply Yes to the verification prompt.

Using MC/ServiceGuard Commands to Start a Package

Use the `cmrunpkg` command to run the package on a particular node, then use the `cmmodpkg` command to enable switching for the package. Example:

```
# cmrunpkg -n ftsys9 pkg1
# cmmodpkg -e pkg1
```

This starts up the package on *ftsys9*, then enables package switching. This sequence is necessary when a package has previously been halted on some node, since halting the package disables switching.

Halting a Package

You halt an MC/ServiceGuard package when you wish to bring the package out of use but wish the node to continue in operation. You can halt a package using SAM or using MC/ServiceGuard commands. Halting a package has a different effect than halting the node. When you halt the node, its packages may switch to adoptive nodes (assuming that switching is enabled for them); when you halt the package, it is disabled from switching to another node, and must be restarted manually on another node or on the same node.

Using SAM to Halt a Package

In the SAM "Package Administration" area, choose a package from the list, then select "Halt Package" from the Actions menu. Choose OK in response to the verification prompt. When you halt the package in this way, it is disabled from switching to another node.

Using MC/ServiceGuard Commands to Halt a Package

Use the `cmhaltpkg` command to halt a package, as follows:

```
# cmhaltpkg pkg1
```

This halts pkg1 and disables it from switching to another node.

Moving a Package

You can use SAM or MC/ServiceGuard commands to move a package from one node to another.

Using SAM to Move a Running Package

From the Package Administration screen in SAM, choose a package, then select "Move a Package" from the Actions menu. Choose the node you wish to move the package to, then select OK. Reply Yes to the verification prompt.

Using MC/ServiceGuard Commands to Move a Running Package

Before you move the package, halt it on its original node using the `cmhaltpkg` command. This action not only halts the package, but also disables switching the package back to the node on which it halts.

After you have moved the package you must restart it and enable switching. You can do this in SAM or by issuing the `cmrunpkg` command followed by `cmmodpkg -e` *package_name*. `cmmodpkg` can be used with the `-n` option to

enable a package to run on a node if the package has been disabled from running on that node due to some sort of error. If no node is specified, the node the command is run on is the implied node.

Example:

```
# cmhaltpkg pkg1
# cmrunpkg -n ftsys10 pkg1
# cmmodpkg -e pkg1
```

Reconfiguring on a Halted Cluster

You can also make permanent changes in package configuration while the cluster is not running. Use the following steps:

- On one node, reconfigure the package as described earlier in this chapter. You can do this by using SAM or by editing the package ASCII file.
- Edit the package control script directly or use the "Edit a Package Control Script" option in SAM. Any changes in service names will also require changes in the package configuration file.
- Use SAM or the `cmapplyconf` command to copy the binary cluster configuration file to all nodes. Use the -P option, specifying the package to be changed; do not use the -C option. This file overwrites any previous version of the binary cluster configuration file.
- Copy the modified control script to all nodes that can run the package.
- Use SAM or the `cmruncl` command to start the cluster on all nodes or on a subset of nodes, as desired. The package will start up as nodes come online.

Reconfiguring a Package on a Running Cluster

You can reconfigure a package while the cluster is running, and in some cases you can reconfigure the package while the package itself is running. Only certain changes may be made while the package is running.

To modify the package, use the SAM "Package Configuration" subarea, and choose options as described in Chapter 6. Alternatively, with HP-UX commands, use the following procedure (*pkg1* is used as an example):

1. Halt the package if necessary:

   ```
   # cmhaltpkg  pkg1
   ```

   See Table 7-2 to determine whether this step is needed.

2. If it is not already available, you can obtain a copy of the package's ASCII configuration file by using the `cmgetconf` command, specifying the package name.

   ```
   # cmgetconf -P pkg1.ascii
   ```

3. Edit the ASCII package configuration file.

4. Verify your changes as follows:

   ```
   # cmcheckconf -v -P pkg1.ascii
   ```

5. Distribute your changes to all nodes:

   ```
   # cmapplyconf -v -P pkg1.ascii
   ```

6. Copy the package control script to all nodes that can run the package.

Adding a Package to a Running Cluster

You can create a new package and add it to the cluster configuration while the cluster is up and while other packages are running. The number of packages you can add is subject to the value of *Maximum Configured Packages* as set in SAM or as defined in the cluster configuration file.

To create the package, follow the steps given in the chapter "Configuring Packages and Services" with the following difference: *do not* specify the cluster name when verifying and distributing the configuration with HP-UX commands. For example, to use HP-UX commands to verify the configuration of newly created *pkg1* on a running cluster:

```
# cmcheckconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

Use an HP-UX command like the following to distribute the new package configuration to all nodes in the cluster:

```
# cmapplyconf -P /etc/cmcluster/pkg1/pkg1conf.ascii
```

Remember to copy the control script to the /etc/cmcluster/pkg1 directory on all nodes that can run the package.

Deleting a Package from a Running Cluster

You can delete a package from all cluster nodes by using the `cmdeleteconf` command. The command can only be executed when the package is not running; the cluster may be up. The command removes the package information from the binary configuration file on all the nodes in the cluster.

The following example halts package *mypkg* and removes the package configuration from the cluster:

```
# cmhaltpkg mypkg
# cmdeleteconf -p mypkg
```

The command prompts for a verification before deleting the files unless you use the `-f` option. The directory `/etc/cmcluster/mypkg` is not deleted by this command.

Changing the `PACKAGE_SWITCHING_ENABLED` Flag

To change package switching behavior in a running cluster so that the new behavior remains in effect across cluster restarts, perform the following steps:

1. To immediately change package switching behavior while the cluster is running, use the `cmmodpkg` command. For example, if you want to disable switching for a running package on a given node, enter the following:

   ```
   # cmmodpkg -d pkg1
   ```

2. To change the default configuration so that the next time the cluster restarts the change you made in package switching is still in effect, you must change the PACKAGE_SWITCHING_ENABLED flag in the package configuration file. (Any change in this flag will only take effect the next time the cluster is restarted.)

   See the previous section "Reconfiguring a Package on a Running Cluster" for detailed instructions.

Allowable Package States During Reconfiguration

All nodes in the cluster must be powered up and accessible when making configuration changes.

Refer to Table 7-2 to determine whether or not the package may be running while you implement a particular kind of change. Note that for all of the following cases the cluster may be running, and also packages other than the one being reconfigured may be running.

Table 7-2 Types of Changes to Packages

Change to the Package	Required Package State
Add a new package	Other packages may be in any state.
Delete a package	Package must not be running.
Add a service	Package must not be running.
Remove a service	Package must not be running.
Add a subnet	Package must not be running. Subnet must already be configured into the cluster.
Remove a subnet	Package must not be running.
Add a resource	Package must not be running. Resource must already be configured into the cluster.
Remove a resource	Package must not be running.
Change run script contents	It is recommended that the package be halted. If the run script for the package is modified while the package is running, timing may cause problems.
Change halt script contents	It is recommended that the package be halted. If the halt script for the package is modified while the package is running, timing may cause problems.
Script timeouts	Package may be either running or halted.
Service timeouts	Package must not be running.
Service failfast	Package must not be running.

Change to the Package	Required Package State
Package switching	Package may be either running or halted.
Network switching	Package may be either running or halted.
Change the order of nodes where a package may run	Package may be either running or halted.

Responding to Cluster Events

MC/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 EMS HA Monitors* (B5735-90001) 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

The number of nodes you will need for your MC/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 MC/ServiceGuard's network failure protection. Even in a multi-node cluster, you could have a situation where all but one node has failed, or where you have shut down all but one node, leaving your cluster in single-node operation. This remaining node will probably have applications running on it. As long as the MC/ServiceGuard daemon *cmcld* is active, other nodes can re-join the cluster at a later time.

If the MC/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 MC/ServiceGuard daemon in a multi-node cluster, which halts the node with a TOC, and causes packages to be switched to adoptive nodes. It is not necessary to halt the single node in this scenario, since the application is still running, and no other node is currently available for package switching. However, you should *not* try to restart MC/ServiceGuard, since data corruption might occur if another node were to attempt to start up a new instance of the application that is still running on the single node.

Instead of restarting the cluster, choose an appropriate time to shutdown and reboot the node, which will allow the applications to shut down and then permit MC/ServiceGuard to restart the cluster after rebooting.

Removing MC/ServiceGuard from a System

If you wish to remove a node from MC/ServiceGuard use, use the `swremove` command to delete the software. Note the following:

- The cluster should not be running on the node from which you will be deleting MC/ServiceGuard.
- The node from which you are deleting MC/ServiceGuard should not be in the cluster configuration.
- If you are removing MC/ServiceGuard from more than one node, **`swremove`** should be issued on one node at a time.

8 Troubleshooting Your Cluster

This chapter describes how to verify cluster operation, how to review cluster status, and how to solve some typical cluster problems. Topics are as follows:

- Testing Cluster Operation
- Monitoring Hardware
- Replacing Disks
- Troubleshooting Approaches
- Solving Problems

Testing Cluster Operation

Once you have configured your MC/ServiceGuard cluster, it's a good idea to 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 SAM

If you have just finished configuring your cluster, start the cluster running before you proceed.

To start the cluster, in SAM:

1. Choose Clusters, then High Availability Clusters.
2. In the Cluster Administration area, highlight the cluster you wish to start, then select "Start Cluster" from the Actions menu.
3. Indicate whether you wish to bring up the cluster on all nodes or on a subset of nodes. If you choose a subset, then you must select the nodes from the list. This option should be used only when you are sure that the cluster is not already running on any node.
4. At the verification prompt, select OK to start the cluster.

Testing the Package Manager

To test that the package manager is operating correctly, perform the following procedure for each package on the cluster:

1. Obtain the PID number of a service in the package by entering

   ```
   # ps -ef | grep "service_cmd"
   ```

   where *service_cmd* is the executable specified in the package control script with the parameter SERVICE_CMD. The service selected must not have restarts specified.

2. To kill the *service_cmd* PID, enter

   ```
   # kill PID
   ```

3. To view the package status, enter

   ```
   # cmviewcl -v
   ```

   The package should be running on the specified adoptive node.

4. Move the package back to the primary node using SAM.

   From the Package Administration screen in SAM, choose a package, then select "Move a Package" from the Actions menu. Choose the node you wish to move the package to, then select OK. Reply Yes to the verification prompt.

   Depending on the specific databases you are running, perform the appropriate database recovery.

Testing the Cluster Manager

To test that the cluster manager is operating correctly, perform the following steps for each node on the cluster:

1. Turn off the power to the node SPU.
2. To observe the cluster reforming, enter the following command on some other configured node:

   ```
   # cmviewcl -v
   ```

   You should be able to observe that the powered down node is halted, and that its packages have been correctly switched to other nodes.

3. Turn on the power to the node SPU.
4. To verify that the node is rejoining the cluster, enter the following command on any configured node:

   ```
   # cmviewcl -v
   ```

   The node should be recognized by the cluster, but its packages should *not* be running.

5. Move the packages back to original node using SAM.

   From the Package Administration screen in SAM, choose a package, then select "Move a Package" from the Actions menu. Choose the node you wish to move the package to, then select OK. Reply Yes to the verification prompt.

   Depending on the specific databases you are running, perform the appropriate database recovery.

6. Repeat this procedure for all nodes in the cluster one at a time.

Testing the Network Manager

To test that the network manager is operating correctly, for each node on the cluster do the following:

1. To identify primary and standby lan cards on the node, enter

   ```
   # lanscan
   ```

   and then

   ```
   # cmviewcl -v
   ```

2. Disconnect the LAN connection from the Primary card. (Be careful not to break the subnet if you are using ThinLAN cables.)
3. Using `cmviewcl -v`, verify that a local switch has taken place so that the Standby card is now the Primary card.
4. Reconnect the LAN to the original Primary card, and verify its status using `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.

Using Event Monitoring Service

Event Monitoring Service (EMS) allows you to configure monitors of specific devices and system resources. You can direct alerts to an administrative workstation where operators can be notified of further action in case of a problem. For example, you could configure a disk monitor to report when a mirror was lost from a mirrored volume group being used in the cluster

Refer to the *Using EMS HA Monitors* (B5735-90001) for additional information.

Using HP Predictive Monitoring

In addition to messages reporting actual device failure, the logs may accumulate messages of lesser severity which, over time, can indicate that a failure may happen soon. One product that provides a degree of automation in monitoring is called HP Predictive, which gathers information from the status queues of a monitored system to see what errors are accumulating. This tool will report failures and will also

predict failures based on statistics for devices that are experiencing specific non-fatal errors over time. In an MC/ServiceGuard cluster, HP Predictive should be run on all nodes.

HP Predictive also reports error conditions directly to an HP Response Center, alerting support personnel to the potential problem. HP Predictive is available through various support contracts. For more information, contact your HP representative.

Replacing Disks

The procedure for replacing a faulty disk mechanism depends on the type of disk configuration you are using. Separate descriptions are provided for replacing an array mechanism and a disk in a high availability enclosure.

Replacing a Faulty Array Mechanism

With any HA disk array configured in RAID 1 or RAID 5, refer to the array's documentation for instruction on how to replace a faulty mechanism. After the replacement, the device itself automatically rebuilds the missing data on the new disk. No LVM activity is needed. This process is known as *hot swapping* the disk.

Replacing a Faulty Mechanism in an HA Enclosure

If you are using software mirroring with MirrorDisk/UX and the mirrored disks are mounted in a high availability disk enclosure, you can use the following steps to *hot plug* a disk mechanism:

1. Identify the physical volume name of the failed disk and the name of the volume group in which it was configured. In the following examples, the volume group name is shown as /dev/vg_sg01 and the physical volume name is shown as /dev/c2t3d0. Substitute the volume group and physical volume names that are correct for your system.
2. Identify the names of any logical volumes that have extents defined on the failed physical volume.
3. On the node on which the volume group is currently activated, use the following command *for each logical volume that has extents on the failed physical volume*:

   ```
   # lvreduce -m 0 /dev/vg_sg01/lvolname /dev/dsk/c2t3d0
   ```

4. At this point, remove the failed disk and insert a new one. The new disk will have the same HP-UX device name as the old one.
5. On the node from which you issued the `lvreduce` command, issue the following command to restore the volume group configuration data to the newly inserted disk:

   ```
   # vgcfgrestore /dev/vg_sg01 /dev/dsk/c2t3d0
   ```

6. Issue the following command to extend the logical volume to the newly inserted disk:

```
# lvextend -m1 /dev/vg_sg01 /dev/dsk/c2t3d0
```

7. Finally, use the `lvsync` command *for each logical volume that has extents on the failed physical volume.* This synchronizes the extents of the new disk with the extents of the other mirror.

```
# lvsync /dev/vg_sg01/lvolname
```

Replacing a Lock Disk

Replacing a failed lock disk mechanism is the same as replacing a data disk. If you are using a *dedicated* lock disk (one with no user data on it), then you need to issue only one LVM command:

```
# vgcfgrestore /dev/vg_lock /dev/dsk/c2t3d0
```

Online Hardware Maintenance with In-line SCSI Terminator

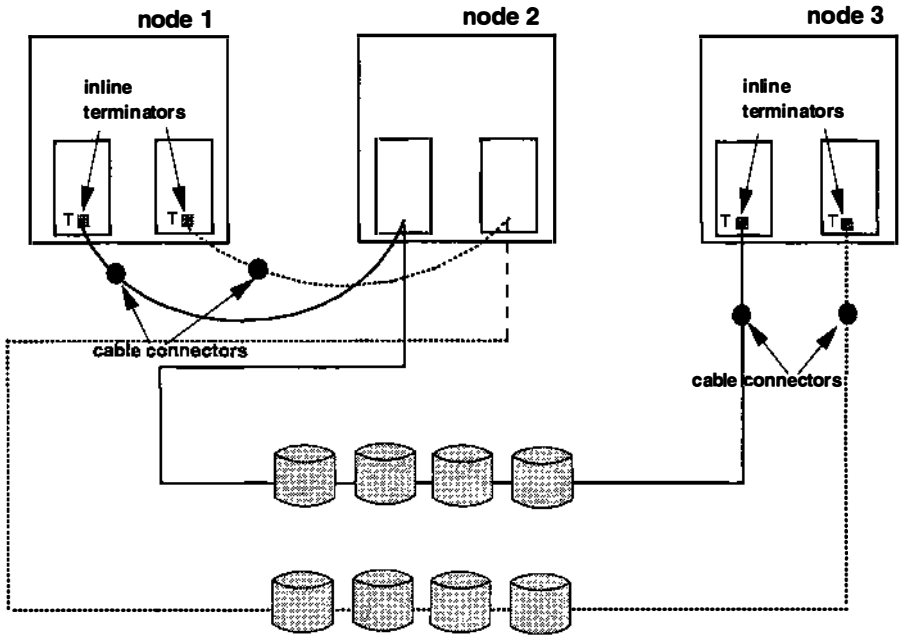
MC/ServiceGuard allows online SCSI disk controller hardware repairs to all cluster nodes if you use HP's *in-line terminator* (C2980A) on nodes connected to the end of the shared FW/SCSI bus. The in-line terminator cable is a 0.5 meter extension cable with the terminator on the male end, which connects to the controller card for an external bus. The in-line terminator is used *instead of* the termination pack that is attached to the controller card and makes it possible to physically disconnect the node from the end of the F/W SCSI bus without breaking the bus's termination. (Nodes attached to the middle of a bus using a Y cable also can be detached from the bus without harm.) When using in-line terminators and Y cables, ensure that all orange-socketed termination packs are *removed* from the controller cards.

NOTE

You cannot use inline terminators with internal FW/SCSI buses on D and K series systems, and you cannot use the inline terminator with single-ended SCSI buses. You must *not* use an inline terminator to connect a node to a Y cable.

Figure 8-1 shows a three-node cluster with two F/W SCSI buses. The solid line and the dotted line represent different buses, both of which have inline terminators attached to nodes 1 and 3. Y cables are also shown attached to node 2.

Figure 8-1 **F/W SCSI Buses with In-line Terminators**



The use of in-line SCSI terminators allows you to do hardware maintenance on a given node by temporarily moving its packages to another node and then halting the original node while its hardware is serviced. Following the replacement, the packages can be moved back to the original node.

Use the following procedure to disconnect a node that is attached to the bus with an in-line SCSI terminator or with a Y cable:

1. Move any packages on the node that requires maintenance to a different node.
2. Halt the node that requires maintenance. The cluster will re-form, and activity will continue on other nodes. Packages on the halted node will switch to other available nodes if they are configured to switch.
3. Disconnect the power to the node.
4. Disconnect the node from the in-line terminator cable or Y cable if necessary. The other nodes accessing the bus will encounter no problems as long as the in-line terminator or Y cable remains connected to the bus.

5. Replace or upgrade hardware on the node, as needed.
6. Reconnect the node to the in-line terminator cable or Y cable if necessary.
7. Reconnect power and reboot the node. If AUTOSTART_CMCLD is set to 1 in the /etc/rc.config.d/cmcluster file, the node will rejoin the cluster.
8. If necessary, move packages back to the node from their alternate locations and restart them.

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 Cluster and Package States
- Reviewing Package IP addresses
- Reviewing the System Log File
- Reviewing Configuration Files
- Reviewing the Package Control Script
- Using `cmquerycl` and `cmcheckconf`
- Using `cmscancl` and `cmgetconf`
- Reviewing the LAN Configuration

Reviewing Cluster and Package States

A cluster or its component nodes may be in several different states at different points in time. Status information for clusters, packages and other cluster elements is shown in the output of the `cmviewcl` command and in some displays in SAM. This section explains the meaning of many of the common conditions the cluster or package may be in.

Information about cluster status is stored in the status database, which is maintained on each individual node in the cluster. You can display information contained in this database by issuing the `cmviewcl` command:

```
# cmviewcl -v
```

The command when issued with the `-v` option displays information about the whole cluster. See the man page for a detailed description of other `cmviewcl` options.

Cluster Status

The *status* of a cluster may be one of the following:

- Up. At least one node has a running cluster daemon, and reconfiguration is not taking place.
- Down. No cluster daemons are running on any cluster node.

- Starting. The cluster is in the process of determining its active membership. At least one cluster daemon is ruuning.

Node Status and State

The *status* of a node is either up (*active* as a member of the cluster) or down (*inactive* in the cluster), depending on whether its cluster daemon is running or not. Note that a node might be down from the cluster perspective, but still up and running HP-UX.

A node may also be in one of the following states:

- Failed. A node never sees itself in this state. Other active members of the cluster will see a node in this state if that node was in an active cluster, but is no longer, and is not halted.
- Cluster Reforming. A node in this state is running the protocols which ensure that all nodes agree to the new membership of an active cluster. If agreement is reached, the status database is updated to reflect the new cluster membership.
- Running. A node in this state has completed all required activity for the last re-formation and is operating normally.
- Halted. A node never sees itself in this state. Other nodes will see it in this state after the node has gracefully left the active cluster, for instance with a `cmhaltnode` command.
- Unknown. A node never sees itself in this state. Other nodes assign a node this state if it has never been an active cluster member.

Package Status and State

The *status* of a package can be one of the following:

- Up. The package control script is active.
- Down. The package control script is not active.
- Unknown.

The *state* of the package can be one of the following:

- Starting. The start instructions in the control script are being run.
- Running. Services are active and being monitored.
- Halting. The halt instructions in the control script are being run.

Packages also have the following switching attributes:

- Package Switching. Enabled means that the package can switch to another node in the event of failure.
- Switching Enabled for a Node. Enabled means that the package can switch to the referenced node. Disabled means that the package cannot switch to the specified node until the node is enabled for the package using the `cmmodpkg` command.

  Every package is marked Enabled or Disabled for each node that is either a primary or adoptive node for the package.

Service Status

Services have only status, as follows:

- Up. The service is being monitored.
- Down. The service is not running. It may have halted or failed.
- Unknown.

Network Status

The network interfaces have only status, as follows:

- Up.
- Down.
- Unknown. We cannot determine whether the interface is up or down. This can happen when the cluster is down. A standby interface has this status.

Serial Line Status

The serial line has only status, as follows:

- Up. Heartbeats are received over the serial line.
- Down. Heartbeat has not been received over the serial line within 2 times the NODE_TIMEOUT value.
- Unknown. We cannot determine whether the serial line is up or down. This can happen when the remote node is down.

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        PKG_SWITCH   NODE
    pkg1          up           running      enabled      ftsys9

      Script_Parameters:
      ITEM          STATUS       NAME        MAX_RESTARTS   RESTARTS
      Service       up           service1               0          0
      Subnet        up           15.13.168.0            0          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        PKG_SWITCH   NODE
    pkg2          up           running      enabled      ftsys10

      Script_Parameters:
      ITEM          STATUS       NAME          MAX_RESTARTS   RESTARTS
      Service       up           service2                 0          0
      Subnet        up           15.13.168.0              0          0

      Node_Switching_Parameters:
      NODE_TYPE     STATUS       SWITCHING    NAME
      Primary       up           enabled      ftsys10      (current)
      Alternate     up           enabled      ftsys9
```

Status After Halting a Package . After halting pkg2 with the `cmhaltpkg` command, the output of `cmviewcl -v` is as follows:

```
CLUSTER       STATUS
example       up

  NODE          STATUS       STATE
  ftsys9        up           running

    Network_Parameters:
    INTERFACE     STATUS       PATH         NAME
    PRIMARY       up           56/36.1      lan0
    STANDBY       up           60/6         lan1
```

```
    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    pkg1         up           running      enabled      ftsys9

      Script_Parameters:
      ITEM          STATUS       NAME          MAX_RESTARTS   RESTARTS
      Service       up           service1                 0          0
      Subnet        up           15.13.168.0              0          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

UNOWNED_PACKAGES

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    pkg2         down         unowned      disabled     unowned

      Node_Switching_Parameters:
      NODE_TYPE     STATUS       SWITCHING     NAME
      Primary       up           enabled       ftsys10
      Alternate     up           enabled       ftsys9
```

Pkg2 now has the status "down", and it is shown as in the unowned state, with package switching disabled. Note that switching is enabled for both nodes, however. This means that once global switching is re-enabled for the package, it will attempt to start up on the primary node.

Status After Moving the Package to Another Node . After issuing the following command:

```
# cmrunpkg -n ftsys9 pkg2
```

the output of the `cmviewcl -v` command is as follows:

```
CLUSTER      STATUS
example      up

  NODE          STATUS       STATE
  ftsys9        up           running

    Network_Parameters:
    INTERFACE     STATUS       PATH          NAME
    PRIMARY       up           56/36.1       lan0
    STANDBY       up           60/6          lan1


    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    pkg1         up           running      enabled      ftsys9

      Script_Parameters:
      ITEM          STATUS       NAME          MAX_RESTARTS   RESTARTS
      Service       up           service1.1               0          0
      Subnet        up           15.13.168.0              0          0

      Node_Switching_Parameters:
      NODE_TYPE     STATUS       SWITCHING     NAME
```

```
    Primary      up           enabled      ftsys9       (current)
    Alternate    up           enabled      ftsys10

  PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
  pkg2         up           running      disabled     ftsys9

    Script_Parameters:
    ITEM         STATUS       NAME          MAX_RESTARTS    RESTARTS
    Service      up           service2.1               0           0
    Subnet       up           15.13.168.0              0           0
    Node_Switching_Parameters:
    NODE_TYPE    STATUS       SWITCHING    NAME
    Primary      up           enabled      ftsys10
    Alternate    up           enabled      ftsys9     (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 it is still disabled from switching.

Status After Package Switching is Enabled . The following command changes package status back to Package Switching Enabled:

```
# cmmodpkg -e pkg2
```

The result is now as follows:

```
CLUSTER      STATUS
example      up

  NODE         STATUS       STATE
  ftsys9       up           running

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    pkg1         up           running      enabled      ftsys9
    pkg2         up           running      enabled      ftsys9

  NODE         STATUS       STATE
  ftsys10      up           running
```

Both packages are now running on ftsys9 and pkg2 is enabled for switching. Ftsys10 is running the daemon and no packages are running on ftsys10.

Status After Halting a Node . After halting *ftsys10*, with the following command:

```
# cmhaltnode  ftsys10
```

the output of `cmviewcl` is as follows on *ftsys9*:

```
CLUSTER      STATUS
example      up

  NODE         STATUS       STATE
  ftsys9       up           running

    PACKAGE      STATUS       STATE        PKG_SWITCH   NODE
    pkg1         up           running      enabled      ftsys9
    pkg2         up           running      enabled      ftsys9
```

```
  NODE          STATUS        STATE
  ftsys10       down          halted
```

This output is seen on both *ftsys9* and *ftsys10*.

Viewing RS232 Status

If you are using a serial (RS232) line as a heartbeat connection, you will see a list of configured RS232 device files in the output of the `cmviewcl -v` command. The following shows normal running status:

```
CLUSTER       STATUS
example       up
  NODE          STATUS        STATE
  ftsys9        up            running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            56/36.1       lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0              up            ftsys10      /dev/tty0p0



  NODE          STATUS        STATE
  ftsys10        up            running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            28.1          lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0              up            ftsys9       /dev/tty0p0
```

The following display shows status after node *ftsys10* has halted:

```
CLUSTER       STATUS
example       up
  NODE          STATUS        STATE
  ftsys9        up            running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            56/36.1       lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0             unknown          ftsys10    /dev/tty0p0



  NODE          STATUS        STATE
  ftsys10       down          running
```

```
    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            28.1          lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0             unknown         ftsys9    /dev/tty0p0
```

The following shows status when the serial line is not working:

```
CLUSTER       STATUS
example       up
  NODE          STATUS        STATE
  ftsys9        up            running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            56/36.1       lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0              down           ftsys10   /dev/tty0p0



  NODE          STATUS        STATE
  ftsys10        up            running

    Network_Parameters:
    INTERFACE     STATUS        PATH          NAME
    PRIMARY       up            28.1          lan0

    Serial_Heartbeat:
    DEVICE_FILE_NAME       STATUS        CONNECTED_TO:
    /dev/tty0p0              down           ftsys9       /dev/tty0p0
```

Reviewing Package IP Addresses

The `netstat -in` command can be used to examine the LAN configuration. The command, if executed on *ftsys9* after the halting of node *ftsys10*, shows that the package IP addresses are assigned to lan0 on *ftsys9* along with the heartbeat IP address.

```
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
ni0*  0     none          none                   0     0        0     0     0
nil*  0     none          none                   0     0        0     0     0
lo0   4608  127           127.0.0.1          10114     0    10114     0     0
lan0  1500  15.13.168     15.13.171.14      959269     2   305189    47 30538
lan0  1500  15.13.168     15.13.171.23      959269     2   305189    47 30538
lan0  1500  15.13.168     15.13.171.20      959269     2   305189    47 30538
lanl* 1500  none          none              418623    27    41716     3  5149
```

Reviewing the System Log File

Messages from the Cluster Manager and Package Manager are written to the system log file. The default location of the log file is `/var/adm/syslog/syslog.log`. You can use a text editor, such as `vi`, or the `more` command to view the log file for historical information on your cluster.

This log provides information on the following:

- Commands executed and their outcome.
- Major cluster events which may, or may not, be errors.
- Cluster status information.

NOTE Many other products running on HP-UX in addition to MC/ServiceGuard use the syslog.log file to save messages. The *HP-UX System Administration Tasks* manual provides additional information on using the system log.

Sample System Log Entries

The following entries from the file */var/adm/syslog/syslog.log* show a package that failed to run due to a problem in the pkg5_run script. You would look at the pkg5_run.log for details.

```
Dec 14 14:33:48 star04 cmcld[2048]: Starting cluster 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 Configuration Files

Review the following ASCII configuration files:

- Cluster configuration file.
- Package configuration files.

Ensure that the files are complete and correct according to your configuration planning worksheets.

Reviewing the Package Control Script

Ensure that the package control script is found on all nodes where the package can run and that the file is identical on all nodes. Ensure that the script is executable on all nodes. Ensure that the name of the control script appears in the package configuration file, and ensure that all services named in the package configuration file also appear in the package control script.

Information about the starting and halting of each package is found in the package's control script log. This log provides the history of the operation of the package control script. It is found at `/etc/cmcluster/`*`package_name`*`/`*`control_script`*`.log`. This log documents all package run and halt activities. If you have written a separate run and halt script script for a package, each script will have its own log.

Using the `cmquerycl` and `cmcheckconf` Commands

In addition, `cmquerycl` and `cmcheckconf` can be used to troubleshoot your cluster just as they were used to verify its configuration. The following example shows the commands used to verify the existing cluster configuration on *ftsys9* and *ftsys10*:

```
# cmquerycl -v -C /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 connectivity.
- The validity of configuration parameters of the cluster and packages for:

- The uniqueness of names.
- The existence and permission of scripts.

It doesn't check:

- The correct setup of the power circuits.
- The correctness of the package configuration script.

Using the `cmscancl` Command

The command `cmscancl` displays information about all the nodes in a cluster in a structured report that allows you to compare such items as IP addresses or subnets, physical volume names for disks, and other node-specific items for all nodes in the cluster. `cmscancl` actually runs several different HP-UX commands on all nodes and gathers the output into a report on the node where you run the command.

The following are the types of configuration data that `cmscancl` displays for each node:

Table 8-1 Data Displayed by the `cmscancl` Command

Description	Source of Data
LAN device configuration and status	`lanscan` command
network status and interfaces	`netstat` command
file systems	`mount` command
LVM configuration	`/etc/lvmtab` file
LVM physical volume group data	`/etc/lvmpvg` file
link level connectivity for all links	`linkloop` command
binary configuration file	`cmviewconf` command

Using the `cmviewconf` Command

`cmviewconf` allows you to examine the binary cluster configuration file, even when the cluster is not running. The command displays the content of this file on the node where you run the command.

Reviewing the LAN Configuration

The following networking commands can be used to diagnose problems:

- `netstat -in` can be used to examine the LAN configuration. This command lists all IP addresses assigned to each LAN interface card.
- `lanscan` can also be used to examine the LAN configuration. This command lists the MAC addresses and status of all LAN interface cards on the node.
- `arp -a` can be used to check the arp tables.
- `landiag` is useful to display, diagnose, and reset LAN card information.
- `linkloop` verifies the communication between LAN cards at MAC address levels. For example, if you enter

  ```
  # linkloop -i4 0x08000993AB72
  ```

  you should see displayed the following message:

  ```
  Link Connectivity to LAN station: 0x08000993AB72  OK
  ```

- `cmscancl` can be used to verify that primary and standby LANs are on the same bridged net.
- `cmviewcl -v` shows the status of primary and standby LANs.

Use these commands on all nodes.

Solving Problems

Problems with MC/ServiceGuard may be of several types. The following is a list of common categories of problem:

- Cluster Re-formations.
- System Administration Errors.
- Package Control Script Hangs.
- Package Movement Errors.
- Node and Network Failures.

The first two categories of problems occur with the incorrect configuration of MC/ServiceGuard. The last category contains "normal" failures to which MC/ServiceGuard is designed to react and ensure the availability of your applications.

Cluster Re-formations

Cluster re-formations may occur from time to time due to current cluster conditions. Some of the causes are as follows:

- local switch on an Ethernet LAN if the switch takes longer than the cluster `NODE_TIMEOUT` value. To prevent this problem, you can increase the cluster `NODE_TIMEOUT` value, or you can use a different LAN type.
- excessive network traffic on heartbeat LANs. To prevent this, you can use dedicated heartbeat LANs, or LANs with less traffic on them.
- an overloaded system, with too much total I/O and network traffic.
- an improperly configured network, for example, one with a very large routing table.

In these cases, applications continue running, though they might experience a small performance impact during cluster re-formation.

System Administration Errors

There are a number of errors you can make when configuring MC/ServiceGuard that will not show up when you start the cluster. Your cluster can be running, and everything appears to be fine, until there is a hardware or software failure and control of your packages are not transferred to another node as you would have expected.

These are errors caused specifically by errors in the cluster configuration file and package configuration scripts. Examples of these errors include:

- Volume groups not defined on adoptive node.
- Mount point does not exist on adoptive node.
- Network errors on adoptive node (configuration errors).
- User information not correct on adoptive node.

You can use the following commands to check the status of your disks:

- `bdf` - to see if your package's volume group is mounted.
- `vgdisplay -v` - to see if all volumes are present.
- `lvdisplay -v` - to see if the mirrors are synchronized.
- `strings /etc/lvmtab` - to ensure that the configuration is correct.
- `ioscan -fnC disk` - to see physical disks.
- `diskinfo -v /dev/rdsk/cxtydz` - to display information about a disk.
- `lssf /dev/dsk/*d0` - to check LV and paths.

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, MC/Serviceguard kills the script and marks the package "Halted." Similarly, when a package control script fails, MC/ServiceGuard kills the script and marks the package "Halted." In both cases, the following also take place:

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

Following such a failure, since the control script is terminated, some of the package's resources may be left activated. Specifically:

- Volume groups may be left active.
- File systems may still be mounted.
- IP addresses may still be installed.
- Services may still be running.

In this kind of situation, MC/Serviceguard will not restart the package without manual intervention. You must clean up manually before restarting the package. Use the following steps as guidelines:

1. Perform application specific cleanup. Any application specific actions the control script might have taken should be undone to ensure successfully starting the package on an alternate node. This might include such things as shutting down application processes, removing lock files, and removing temporary files.
2. Ensure that package IP addresses are removed from the system. This step is accomplished via the `cmmodnet (1m)` command. First determine which package IP addresses are installed by inspecting the output resulting from running the command `netstat -in`. If any of the IP addresses specified in the package control script appear in the netstat output under the "Address" column, use `cmmodnet` to remove them:

   ```
   # cmmodnet -r -i <ip-address> <subnet>
   ```

   where <ip-address> is the address in the "Address" column and <subnet> is the corresponding entry in the "Network" column.
3. Ensure that package volume groups are deactivated. First unmount any package logical volumes which are being used for filesystems. This is determined by inspecting the output resulting from running the command `bdf -l`. If any package logical volumes, as specified by the `LV[]` array variables in the package control script, appear under the "Filesystem" column, use `umount` to unmount them:

   ```
   # fuser -ku <logical-volume>
   # umount <logical-volume>
   ```

   Next, deactivate the package volume groups. These are specified by the `VG[]` array entries in the package control script.

   ```
   # vgchange -a n <volume-group>
   ```
4. Finally, re-enable the package for switching.

   ```
   # cmmodpkg -e <package-name>
   ```

If after cleaning up the node on which the timeout occured it is desirable to have that node as an alternate for running the package, remember to re-enable the package to run on the node:

```
# cmmodpkg -e -n <node-name> <package-name>
```

The default ServiceGuard control scripts are designed to take the straightforward steps needed to get an application running or stopped. If the package administrator specifies a time limit within which these steps need to occur and that limit is subsequently exceeded for any reason, ServiceGuard takes the conservative approach that the control script logic must either be hung or defective in some way. At that point the control script cannot be trusted to perform cleanup actions correctly, thus the script is terminated and the package administrator is given the opportunity to assess what cleanup steps must be taken.

If you want the package to switch automatically in the event of a control script timeout, set the `NODE_FAIL_FAST_ENABLED` parameter to `YES`. (If you are using SAM, set *Package Failfast* to *Enabled*.) In this case, ServiceGuard will cause a TOC on the node where the control script timed out. This effectively cleans up any side effects of the package's run or halt attempt. In this case the package will be automatically restarted on any available alternate node for which it is configured.

Package Movement Errors

These errors are similar to the system administration errors except they are caused specifically by errors in the package control script. The best way to prevent these errors is to test your package control script before putting your high availability application on line.

Running your script with the `-x` shell option will give you details on where your script may be failing.

Node and Network Failures

These failures cause MC/ServiceGuard to transfer control of a package to another node. This is the normal action of MC/ServiceGuard, but you have to be able to recognize when a transfer has taken place and decide to leave the cluster is its current condition or to restore it to its original condition.

Possible node failures can be caused by the following conditions:

- HPMC.
- 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.
- `landiag` - to display, test, and reset the LAN cards.

Since your cluster is unique, there are no cookbook solutions to 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.

A MC/ServiceGuard Commands

The following is a list of commands used for MC/ServiceGuard configuration and maintenance. Man pages for these commands are available on your system *after installation.*

man page	description
cmapplyconf	Verify and apply MC/ServiceGuard and MC/LockManager cluster configuration and package configuration files. cmapplyconf verifies the cluster configuration and package configuration specified in the cluster_ascii_file and the associated pkg_ascii_file(s) , creates or updates the binary configuration file, called cmclconfig, and distributes it to all nodes. This binary configuration file contains the cluster configuration information as well as package configuration information for all packages specified. This file, which is used by the cluster daemons to manage the entire cluster and package environment, is kept in the /etc/cmcluster directory. If changes to either the cluster configuration or to any of the package configuration files are needed, first update the appropriate ASCII ile(s) (cluster or package), then validate the changes using the cmcheckconf command and then use cmapplyconf again to verify and redistribute the binary file to all nodes. The cluster and package configuration can be modified whether the cluster is up or down, although some configuration requires either the cluster or the package be halted. Please refer to the manual for more detail. The cluster ASCII file only needs to be specified if configuring the cluster for he first time, or if adding or deleting nodes to the cluster. The package ASCII file only needs to be specified if the package is being added, or if the package configuration is being modified. It is recommended that the user run the cmgetconf command to get either the cluster ASCII configuration file or package ASCII configuration file whenever changes to the existing configuration are required. Note that cmapplyconf will verify and distribute cluster configuration or package files. It will not cause the cluster daemon to start or removed from the cluster configuration. The same kind of processing will apply to the package configuration to determine whether to add or delete package nodes, package subnet, etc. Not all package configuration changes require the package to be halted.

man page	description
cmcheckconf	Check high availability cluster configuration and/or package configuration files. cmcheckconf verifies the cluster configuration as specified by the cluster_ascii_file and/or the package configuration files specified by each pkg_ascii_file in the command. If the cluster has already been configured previously, the cmcheckconf command will compare the configuration in the cluster_ascii_file against the previously configuration information stored in the binary configuration file and validates the changes. The same rules apply to the pkg_ascii_file. It is not necessary to halt either the cluster or any of the packages to run the cmcheckconf command.
cmdeleteconf	Delete either the cluster or the package configuration. cmdeleteconf deletes either the entire cluster configuration, including all its packages, or only the specified package configuration. If neither cluster_name nor package_name is specified, cmdeleteconf will delete the local cluster's configuration and all its packages. If only the package_name is specified, the configuration of package_name in the local cluster is deleted. If both cluster_name and package_name are specified, the package must be configured in the cluster_name, and only the package package_name will be deleted. The local cluster is the cluster that the node running the cmdeleteconf command belongs to.
cmgetconf	Get cluster or package configuration information. Cmgetconf obtains either the cluster configuration, not including the package configuration, or the specified package's configuration information, and writes to either the output_filename file, or to stdout. This command can be run whether the cluster is up or down. If neither cluster_name nor package_name is specified, cmgetconf will obtain the local cluster's configuration. If both cluster_name and package_name are specified, the package must be configured in the cluster_name, and only the package configuration for package_name will be written to output_filename or to stdout.

man page	description
cmhaltcl	Halt a high availability cluster. cmhaltcl causes all nodes in a configured cluster to stop their cluster daemons, optionally halting all packages or applications in the process. This command will halt all the daemons on all currently running systems. If the user only wants to shutdown a subset of daemons, the cmhaltnode command should be used instead.
cmhaltnode	Halt a node in a high availability cluster. cmhaltnode causes a node to halt its cluster daemon and remove itself from the existing cluster. When cmhaltnode is run on a node, the cluster daemon is halted and, optionally, all packages that were running on that node are moved to other nodes if possible.
cmhaltpkg	Halt a high availability package. cmhaltpkg performs a manual halt of high availability package(s) running on MC/ServiceGuard or MC/LockManager clusters. This command may be run on any node within the cluster and may operate on any package within the cluster.
cmmodpkg	Enable or disable switching attributes for a high availability package. cmmodpkg enables or disables the ability of a package to switch to another node upon failure of the package, and it enables or disables a particular node from running specific packages.Switching for a package can be enabled or disabled globally. For example, if a globally disabled package fails, it will not switch to any other node, and if a globally enabled package fails, it will attempt to switch to the first available node on which it is configured to run.

man page	description
cmquerycl	Query cluster or node configuration information. cmquerycl searches all specified nodes for cluster configuration and Logical Volume Manager (LVM) information. Cluster configuration information includes network information such as LAN interface, IP addresses, bridged networks and possible heartbeat networks. LVM information includes volume group (VG) interconnection and file systemmount point information. This command should be run as the first step in preparing for cluster configuration. It may also be used as a troubleshooting tool to identify the current configuration of a cluster.
cmruncl	Run a high availability cluster. cmruncl causes all nodes in a configured cluster or all nodes specified to start their cluster daemons and form a new cluster.This command should only be run when the cluster is not active on any of the configured nodes. If a cluster is already running on a subset of the nodes, the cmrunnode command should be used to start the remaining nodes and force them to join the existing cluster.
cmrunnode	Run a node in a high availability cluster. cmrunnode causes a node to start its cluster daemon to join the existing cluster. Starting a node will not cause any active packages to be moved to the new node. However, if a package is DOWN, has its switching enabled, and is able to run on the new node, that package will automatically run there.

man page	description
cmrunpkg	Run a high availability package. cmrunpkg runs a high availability package(s) that was previously halted.This command may be run on any node within the cluster and may operate on any package within the cluster. If a node is not specified, the node on which the command is run will be used. This will result in an error if the current node is not able to run the package or is not in the list of possible owners of the package. When a package is started on a new node, the package's run script is executed.
cmscancl	Gather system configuration information from nodes with MC/ServiceGuard or MC/LockManager installed. cmscancl is a configuration report and diagnostic tool which gathers system software and hardware configuration information from a list of nodes, or from all the nodes in a cluster. The information that this command displays includes LAN device configuration, network status and interfaces, file systems, LVM configuration, link-level connectivity, and the data from the binary cluster configuration file. This command can be used as a troubleshooting tool or as a data collection tool.
cmviewcl	View information about the current high availability cluster. cmviewcl displays the current status information of a cluster. Output can be displayed for the whole cluster or it may be limited to particular nodes or packages.
cmviewconf	View MC/ServiceGuard or MC/LockManager cluster configuration information. cmviewconf collects and displays the cluster configuration information, in ASCII format, from the binary configuration file for an existing cluster. Optionally, the output can be written to a file. This command can be used as a troubleshooting tool to identify the configuration of a cluster.

B Enterprise Cluster Master Toolkit

The Enterprise Cluster Master Toolkit (HP Product Number B5121) provides a group of example scripts and package configuration files for creating MC/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.

Toolkits are included for the following HP Domain server components:

- FastTrack Server
- Enterprise Server
- Enterprise Server Pro
- Proxy Server

Toolkits are included for the following database products:

- Informix
- Oracle
- Progress
- Sybase

A separate NFS toolkit is available. Refer to *Managing Highly Available NFS* (HP Part Number B5125-90001) for more information.

Numerous other application integration scripts are available from your HP representative.

C Designing Highly Available Cluster Applications

This appendix describes how to create or port applications for high availability, with emphasis on the following topics:

- Automating Application Operation
- Controlling the Speed of Application Failover
- Designing Applications to Run on Multiple Systems
- Restoring Client Connections
- Handling Application Failures
- Minimizing Planned Downtime

Designing for high availability means reducing the amount of unplanned and planned downtime that users will experience. Unplanned downtime includes unscheduled events such as power outages, system failures, network failures, disk crashes, or application failures. Planned downtime includes scheduled events such as scheduled backups, system upgrades to new OS revisions, or hardware replacements.

Two key strategies should be kept in mind:

1. Design the application to handle a system reboot or panic. If you are modifying an existing application for a highly available environment, determine what happens currently with the application after a system panic. In a highly available environment there should be defined (and scripted) procedures for restarting the application. Procedures for starting and stopping the application should be automatic, with no user intervention required.
2. The application should not use any system-specific information such as the following if such use would prevent it from failing over to another system and running properly:
   - The application should not refer to `uname()` or `gethostname()`.
   - The application should not refer to the SPU ID.
   - The application should not refer to the MAC (link-level) address.

Automating Application Operation

Can the application be started and stopped automatically or does it require operator intervention?

This section describes how to automate application operations to avoid the need for user intervention. One of the first rules of high availability is to avoid manual intervention. If it takes a user at a terminal, console or GUI interface to enter commands to bring up a subsystem, the user becomes a key part of the system. It may take hours before a user can get to a system console to do the work necessary. The hardware in question may be located in a far-off area where no trained users are available, the systems may be located in a secure datacenter, or in off hours someone may have to connect via modem.

There are two principles to keep in mind for automating application relocation:

- Insulate users from outages.
- Applications must have defined startup and shutdown procedures.

You need to be aware of what happens currently when the system your application is running on is rebooted, and whether changes need to be made in the application's response for high availability.

Insulate Users from Outages

Wherever possible, insulate your end users from outages. Issues include the following:

- Do not require user intervention to reconnect when a connection is lost due to a failed server.
- Where possible, warn users of slight delays due to a failover in progress.
- Minimize the reentry of data.
- Engineer the system for reserve capacity to minimize the performance degradation experienced by users.

Define Applications' Startup and Shutdown

Applications must be restartable without manual intervention. If the application requires a switch to be flipped on a piece of hardware, then automated restart is impossible. Procedures for application startup, shutdown and monitoring must be created so that the HA software can perform these functions automatically.

To ensure automated response, there should be defined procedures for starting up the application and stopping the application. In MC/ServiceGuard these procedures are placed in the package control script. These procedures must check for errors and return status to the HA control software. The startup and shutdown should be command-line driven and not interactive unless all of the answers can be predetermined and scripted.

In an HA failover environment, HA software restarts the application on a surviving system in the cluster that has the necessary resources, like access to the necessary disk drives. The application must be restartable in two aspects:

- It must be able to restart and recover on the backup system (or on the same system if the application restart option is chosen).
- It must be able to restart if it fails during the startup and the cause of the failure is resolved.

Application administrators need to learn to startup and shutdown applications using the appropriate HA commands. Inadvertently shutting down the application directly will initiate an unwanted failover. Application administrators also need to be careful that they don't accidently shut down a production instance of an application rather than a test instance in a development environment.

A mechanism to monitor whether the application is active is necessary so that the HA software knows when the application has failed. This may be as simple as a script that issues the command `ps -ef | grep xxx` for all the processes belonging to the application.

To reduce the impact on users, the application should not simply abort in case of error, since aborting would cause an unneeded failover to a backup system. Applications should determine the exact error and take specific action to recover from the error rather than, for example, aborting upon receipt of any error.

Controlling the Speed of Application Failover

What steps can be taken to ensure the fastest failover?

If a failure does occur causing the application to be moved (failed over) to another node, there are many things the application can do to reduce the amount of time it takes to get the application back up and running. The topics covered are as follows:

- Replicate Non-Data File Systems
- Use Raw Volumes
- Evaluate the Use of JFS
- Minimize Data Loss
- Use Restartable Transactions
- Use Checkpoints
- Design for Multiple Servers
- Design for Replicated Data Sites

Replicate Non-Data File Systems

Non-data file systems should be replicated rather than shared. There can only be one copy of the application data itself. It will be located on a set of disks that is accessed by the system that is running the application. After failover, if these data disks are filesystems, they must go through filesystems recovery (`fsck`) before the data can be accessed. To help reduce this recovery time, the smaller these filesystems are, the faster the recovery will be. Therefore, it is best to keep anything that can be replicated off the data filesystem. For example, there should be a copy of the application executables on each system rather than having one copy of the executables on a shared filesystem. Additionally, replicating the application executables makes them subject to a rolling upgrade if this is desired.

Use Raw Volumes

If your application uses data, use raw volumes rather than filesystems. Raw volumes do not require an `fsck` of the filesystem, thus eliminating one of the potentially lengthy steps during a failover.

Evaluate the Use of JFS

If a file system must be used, a JFS offers significantly faster file system recovery as compared to an HFS. However, performance of the JFS may vary with the application.

Minimize Data Loss

Minimize the amount of data that might be lost at the time of an unplanned outage. It is impossible to prevent some data from being lost when a failure occurs. However, it is advisable to take certain actions to minimize the amount of data that will be lost, as explained in the following discussion.

Minimize the Use and Amount of Memory-Based Data

Any in-memory data (the in-memory context) will be lost when a failure occurs. The application should be designed to minimize the amount of in-memory data that exists unless this data can be easily recalculated. When the application restarts on the standby node, it must recalculate or reread from disk any information it needs to have in memory.

One way to measure the speed of failover is to calculate how long it takes the application to start up on a normal system after a reboot. Does the application start up immediately? Or are there a number of steps the application must go through before an end-user can connect to it? Ideally, the application can start up quickly without having to reinitialize in-memory data structures or tables.

Performance concerns might dictate that data be kept in memory rather than written to the disk. However, the risk associated with the loss of this data should be weighed against the performance impact of posting the data to the disk.

Data that is read from a shared disk into memory, and then used as read-only data can be kept in memory without concern.

Keep Logs Small

Some databases permit logs to be buffered in memory to increase online performance. Of course, when a failure occurs, any in-flight transaction will be lost. However, minimizing the size of this in-memory log will reduce the amount of completed transaction data that would be lost in case of failure.

Keeping the size of the on-disk log small allows the log to be archived or replicated more frequently, reducing the risk of data loss if a disaster were to occur. There is, of course, a trade-off between online performance and the size of the log.

Eliminate Need for Local Data

When possible, eliminate the need for local data. In a three-tier, client/server environment, the middle tier can often be dataless (i.e., there is no local data that is client specific or needs to be modified). This "application server" tier can then provide additional levels of availability, load-balancing, and failover. However, this scenario requires that all data be stored either on the client (tier 1) or on the database server (tier 3).

Use Restartable Transactions

Transactions need to be restartable so that the client does not need to re-enter or back out of the transaction when a server fails, and the application is restarted on another system. In other words, if a failure occurs in the middle of a transaction, there should be no need to start over again from the beginning. This capability makes the application more robust and reduces the visibility of a failover to the user.

A common example is a print job. Printer applications typically schedule jobs. When that job completes, the scheduler goes on to the next job. If, however, the system dies in the middle of a long job (say it is printing paychecks for 3 hours), what happens when the system comes back up again? Does the job restart from the beginning, reprinting all the paychecks, does the job start from where it left off, or does the scheduler assume that the job was done and not print the last hours worth of paychecks? The correct behavior in a highly available environment is to restart where it left off, ensuring that everyone gets one and only one paycheck.

Another example is an application where a clerk is entering data about a new employee. Suppose this application requires that employee numbers be unique, and that after the name and number of the new employee is entered, a failure occurs. Since the employee number had been entered before the failure, does the application refuse to allow it to be re-entered? Does it require that the partially entered information be deleted first? More appropriately, in a highly available environment the application will allow the clerk to easily restart the entry or to continue at the next data item.

Use Checkpoints

Design applications to checkpoint complex transactions. A single transaction from the user's perspective may result in several actual database transactions. Although this issue is related to restartable transactions, here it is advisable to record progress locally on the client so that a transaction that was interrupted by a system failure can be completed after the failover occurs.

For example, suppose the application being used is calculating PI. On the original system, the application has gotten to the 1,000th decimal point, but the application has not yet written anything to disk. At that moment in time, the node crashes. The application is restarted on the second node, but the application is started up from scratch. The application must recalculate those 1,000 decimal points. However, if the application had written to disk the decimal points on a regular basis, the application could have restarted from where it left off.

Balance Checkpoint Frequency with Performance

It is important to balance checkpoint frequency with performance. The trade-off with checkpointing to disk is the impact of this checkpointing on performance. Obviously if you checkpoint too often the application slows; if you don't checkpoint often enough, it will take longer to get the application back to its current state after a failover. Ideally, the end-user should be able to decide how often to checkpoint. Applications should provide customizable parameters so the end-user can tune the checkpoint frequency.

Design for Multiple Servers

If you use multiple active servers, multiple service points can provide relatively transparent service to a client. However, this capability requires that the client be smart enough to have knowledge about the multiple servers and the priority for addressing them. It also requires access to the data of the failed server or replicated data.

For example, rather than having a single application which fails over to a second system, consider having both systems running the application. After a failure of the first system, the second system simply takes over the load of the first system. This eliminates the start up time of the application. There are many ways to design this sort of architecture, and there are also many issues with this sort of design. This discussion will not go into details other than to give a few examples.

The simplest method is to have two applications running in a master/slave relationship where the slave is simply a hot standby application for the master. When the master fails, the slave on the second system would still need to figure out what state the data was in (i.e., data recovery would still take place). However, the time to fork the application and do the initial startup is saved.

Another possibility is having two applications that are both active. An example might be two application servers which feed a database. Half of the clients connect to one application server and half of the clients connect to the second application server. If one server fails, then all the clients connect to the remaining application server.

Design for Replicated Data Sites

Replicated data sites are a benefit for both fast failover and disaster recovery. With replicated data, data disks are *not* shared between systems. There is no data recovery that has to take place. This makes the recovery time faster. However, there may be performance trade-offs associated with replicating data. There are a number of ways to perform data replication, which should be fully investigated by the application designer.

Many of the standard database products provide for data replication transparent to the client application. By designing your application to use a standard database, the end-user can determine if data replication is desired.

Designing Applications to Run on Multiple Systems

If an application can be failed to a backup node, how will it work on that different system?

The previous sections discussed methods to ensure that an application can be automatically restarted. This section will discuss some ways to ensure the application can run on multiple systems. Topics are as follows:

- Avoid Node Specific Information
- Assign Unique Names to Applications
- Use `Uname(2)` With Care
- Bind to a Fixed Port
- Bind to a Relocatable IP Addresses
- Give Each Application its Own Volume Group
- Use Multiple Destinations for SNA Applications
- Avoid File Locking

Avoid Node Specific Information

Typically, when a new system is installed, an IP address must be assigned to each active network interface. This IP address is always associated with the node and is called a **stationary** IP address.

The use of packages containing highly available applications adds the requirement for an additional set of IP addresses, which are assigned to the applications themselves. These are known as **relocatable** application IP addresses. MC/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 MC/ServiceGuard, the associated subnetwork address is specified as a package dependency, and a list of nodes on which the package can run is also provided. When failing a package over to a remote node, the subnetwork must already be active on the target node.

Each application or package should be given a unique name as well as a relocatable IP address. Following this rule separates the application from the system on which it runs, thus removing the need for user knowledge of which system the application

runs on. It also makes it easier to move the application among different systems in a cluster for for load balancing or other reasons. If two applications share a single IP address, they must move together. Instead, using independent names and addresses allows them to move separately.

For external access to the cluster, clients must know how to refer to the application. One option is to tell the client which relocatable IP address is associated with the application. Another option is to think of the application name as a host, and configure a name-to-address mapping in the Domain Name System (DNS). In either case, the client will ultimately be communicating via the application's relocatable IP address. If the application moves to another node, the IP address will move with it, allowing the client to use the application without knowing its current location. Remember that each network interface must have a stationary IP address associated with it. This IP address does *not* move to a remote system in the event of a network failure.

Obtain Enough IP Addresses

Each application receives a *relocatable* IP address that is separate from the stationary IP address assigned to the system itself. Therefore, a single system might have many IP addresses, one for itself and one for each of the applications that it normally runs. Therefore, IP addresses in a given subnet range will be consumed faster than without high availablity. It might be necessary to acquire additional IP addresses.

Multiple IP addresses on the same network interface are supported only if they are on the same subnetwork.

Allow Multiple Instances on Same System

Applications should be written so that multiple instances, each with its own application name and IP address, can run on a single system. It might be necessary to invoke the application with a parameter showing which instance is running. This allows distributing the users among several systems under normal circumstances, but it also allows all of the users to be serviced in the case of a failure on a single system.

Avoid Using SPU IDs or MAC Addresses

Design the application so that it does not rely on the SPU ID or MAC (link-level) addresses. The SPU ID is a unique hardware ID contained in non-volatile memory, which cannot be changed. A MAC address (also known as a LANIC id) is a link-specific address associated with the LAN hardware. The use of these addresses is a common problem for license servers, since for security reasons they want to use

hardware-specific identification to ensure the license isn't copied to multiple nodes. One workaround is to have multiple licenses; one for each node the application will run on. Another way is to have a cluster-wide mechanism that lists a set of SPU IDs or node names. If your application is running on a system in the specified set, then the license is approved.

Previous generation HA software would move the MAC address of the network card along with the IP address when services were moved to a backup system. This is no longer allowed in MC/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 relocatable IP addresses.

Instead, the application should always refer to the application name and relocatable IP address rather than the hostname and stationary IP address. It is appropriate for the application to call `gethostbyname(2)`, specifying the application name rather than the hostname. `gethostbyname(2)` will pass in the IP address of the application. This IP address will move with the application to the new node.

However, `gethostbyname(2)` should be used to locate the IP address of an application only if the application name is configured in DNS. It is probably best to associate a different application name with each independent HA service. This allows each application and its IP address to be moved to another node without affecting other applications. Only the stationary IP addresses should be associated with the hostname in DNS.

Use uname(2) With Care

Related to the hostname issue discussed in the previous section is the application's use of `uname(2)`, which returns the official system name. The system name is unique to a given system whatever the number of LAN cards in the system. By convention, the `uname` and `hostname` are the same, but they do not have to be. Some applications, after connection to a system, might call `uname(2)` to validate for security purposes that they are really on the correct system. This is not appropriate in an HA environment, since the service is moved from one system to another, and neither the `uname` nor the `hostname` are moved. Applications should develop alternate means of verifying where they are running. For example, an application might check a list of hostnames that have been provided in a configuration file.

Bind to a Fixed Port

When binding a socket, a port address can be specified or one can be assigned dynamically. One issue with binding to random ports is that a different port may be assigned if the application is later restarted on another cluster node. This may be confusing to clients accessing the application.

The recommended method is using fixed ports that are the same on all nodes where the application will run, instead of assigning port numbers dynamically. The application will then always return the same port number regardless of which node is currently running the application. Application port assignments should be put in `/etc/services` to keep track of them and to help ensure that someone will not choose the same port number.

Bind to Relocatable IP Addresses

When sockets are bound, an IP address is specified in addition to the port number. This indicates the IP address to use for communication and is meant to allow applications to limit which interfaces can communicate with clients. An application can bind to INADDR_ANY as an indication that messages can arrive on any interface.

Network applications can bind to a stationary IP address, a relocatable IP address, or INADDR_ANY. If the stationary IP address is specified, then the application may fail when restarted on another node, because the stationary IP address is not moved to the new system. If an application binds to the relocatable IP address, then the application will behave correctly when moved to another system.

Many server-style applications will bind to INADDR_ANY, meaning that they will receive requests on any interface. This allows clients to send to the stationary or relocatable IP addresses. However, in this case the networking code cannot determine which source IP address is most appropriate for responses, so it will always pick the stationary IP address.

For TCP stream sockets, the TCP level of the protocol stack resolves this problem for the client since it is a connection-based protocol. On the client, TCP ignores the stationary IP address and continues to use the previously bound relocatable IP address originally used by the client.

With UDP datagram sockets, however, there is a problem. The client may connect to multiple servers utilizing the relocatable IP address and sort out the replies based on the source IP address in the server's response message. However, the source IP address given in this response will be the stationary IP address rather than the relocatable application IP address. Therefore, when creating a UDP socket for listening, the application must always call `bind(2)` with the appropriate relocatable application IP address rather than `INADDR_ANY`.

If the application cannot be modified as recommended above, a workaround to this problem is to not use the stationary IP address at all, and only use a single relocatable application IP address on a given LAN card. Limitations with this workaround are as follows:

- Local LAN failover will not work.
- There has to be an idle LAN card on each backup node that is used to relocate the relocatable application IP address in case of a failure.

Call bind() before connect()

When an application initiates its own connection, it should first call `bind(2)`, specifying the application IP address before calling `connect(2)`. Otherwise the connect request will be sent using the stationary IP address of the system's outbound LAN interface rather than the desired relocatable application IP address. The client will receive this IP address from the `accept(2)` call, possibly confusing the client software and preventing it from working correctly.

Give Each Application its Own Volume Group

Use separate volume groups for each application that uses data. If the application doesn't use disk, it is not necessary to assign it a separate volume group. A volume group (group of disks) is the unit of storage that can move between nodes. The greatest flexibility for load balancing exists when each application is confined to its own volume group, i.e., two applications do not share the same set of disk drives. If two applications do use the same volume group to store their data, then the applications must move together. If the applications' data stores are in separate volume groups, they can switch to different nodes in the event of a failover.

The application data should be set up on different disk drives and if applicable, different mount points. The application should be designed to allow for different disks and separate mount points. If possible, the application should not assume a specific mount point.

To prevent one node from inadvertently accessing disks being used by the application on another node, HA software uses an exclusive access mechanism to enforce access by only one node at a time. This exclusive access applies to a volume group as a whole.

Use Multiple Destinations for SNA Applications

SNA is point-to-point link-oriented; that is, the *services* cannot simply be moved to another system, since that system has a different point-to-point link which originates in the mainframe. Therefore, backup links in a node and/or backup links in other nodes should be configured so that SNA does not become a single point of failure. Note that only one configuration for an SNA link can be active at a time. Therefore, backup links that are used for other purposes should be reconfigured for the primary mission-critical purpose upon failover.

Avoid File Locking

In an NFS environment, applications should avoid using file-locking mechanisms, where the file to be locked is on an NFS Server. File locking should be avoided in an application both on local and remote systems. If local file locking is employed and the system fails, the system acting as the backup system will not have any knowledge of the locks maintained by the failed system. This may or may not cause problems when the application restarts.

Remote file locking is the worst of the two situations, since the system doing the locking may be the system that fails. Then, the lock might never be released, and other parts of the application will be unable to access that data. In an NFS environment, file locking can cause long delays in case of NFS client system failure and might even delay the failover itself.

Restoring Client Connections

How does a client reconnect to the server after a failure?

It is important to write client applications to specifically differentiate between the loss of a connection to the server and other application-oriented errors that might be returned. The application should take special action in case of connection loss.

One question to consider is how a client knows after a failure when to reconnect to the newly started server. The typical scenario is that the client must simply restart their session, or relog in. However, this method is not very automated. For example, a well-tuned hardware and application system may fail over in 5 minutes. But if users, after experiencing no response during the failure, give up after 2 minutes and go for coffee and don't come back for 28 minutes, the perceived downtime is actually 30 minutes, not 5. Factors to consider are the number of reconnection attempts to make, the frequency of reconnection attempts, and whether or not to notify the user of connection loss.

There are a number of strategies to use for client reconnection:

- Design clients which continue to try to reconnect to their failed server.

  Put the work into the client application rather than relying on the user to reconnect. If the server is back up and running in 5 minutes, and the client is continually retrying, then after 5 minutes, the client application will reestablish the link with the server and either restart or continue the transaction. No intervention from the user is required.

- Design clients to reconnect to a *different* server.

  If you have a server design which includes multiple active servers, the client could connect to the second server, and the user would only experience a brief delay.

  The problem with this design is knowing when the client should switch to the second server. How long does a client retry to the first server before giving up and going to the second server? There are no definitive answers for this. The answer depends on the design of the server application. If the application can be restarted on the same node after a failure (see "Handling Application Failures" following), the retry to the current server should continue for the amount of time it takes to restart the server locally. This will keep the client from having to switch to the second server in the event of a application failure.

- Use a transaction processing monitor or message queueing software to increase robustness.

Use transaction processing monitors such as Tuxedo or DCE/Encina, which provide an interface between the server and the client. Transaction processing monitors (TPMs) can be useful in creating a more highly available application. Transactions can be queued such that the client does not detect a server failure. Many TPMs provide for the optional automatic rerouting to alternate servers or for the automatic retry of a transaction. TPMs also provide for ensuring the reliable completion of transactions, although they are not the only mechanism for doing this. After the server is back online, the transaction monitor reconnects to the new server and continues routing it the transactions.

- Queue Up Requests

  As an alternative to using a TPM, queue up requests when the server is unavailable. Rather than notifying the user when a server is unavailable, the user request is queued up and transmitted later when the server becomes available again. Message queueing software ensures that messages of any kind, not necessarily just transactions, are delivered and acknowledged.

  Message queueing is useful only when the user does not need or expect response that the request has been completed (i.e, the application is not interactive).

Handling Application Failures

What happens if part or all of an application fails?

All of the preceding sections have assumed the failure in question was not a failure of the application, but of another component of the cluster. This section deals specifically with application problems. For instance, software bugs may cause an application to fail or system resource issues (such as low swap/memory space) may cause an application to die. The section deals with how to design your application to recover after these types of failures.

Create Applications to be Failure Tolerant

An application should be tolerant of failures in a single component. Many applications have multiple processes running on a single node. If one process fails, what happens to the other processes? Do they also fail? Can the failed process be restarted on the same node without affecting the remaining pieces of the application?

Ideally, if one process fails, the other processes can wait a period of time for that component to come back online. This is true whether the component is on the same system or a remote system. The failed component can be restarted automatically on the same system and rejoin the waiting processing and continue on. This type of failure can be detected and restarted within a few seconds, so the end user would never know a failure occurred.

Another alternative is for the failure of one component to still allow bringing down the other components cleanly. If a database SQL server fails, the database should still be able to be brought down cleanly so that no database recovery is necessary.

The worse case is for a failure of one component to cause the entire system to fail. If one component fails and all other components need to be restarted, the downtime will be high.

Be Able to Monitor Applications

All components in a system, including applications, should be able to be monitored for their health. A monitor might be as simple as a display command or as complicated as a SQL query. There must be a way to ensure that the application is behaving correctly. If the application fails and it is not detected automatically, it might take hours for a user to determine the cause of the downtime and recover from it.

Minimizing Planned Downtime

Planned downtime (as opposed to unplanned downtime) is scheduled; examples include backups, systems upgrades to new operating system revisions, or hardware replacements. For planned downtime, application designers should consider:

- **Reducing the time needed for application upgrades/patches.**

  Can an administrator install a new version of the application without scheduling downtime? Can different revisions of an application operate within a system? Can different revisions of a client and server operate within a system?

- **Providing for online application reconfiguration.**

  Can the configuration information used by the application be changed without bringing down the application?

- **Documenting maintenance operations.**

  Does an operator know how to handle maintenance operations?

When discussing highly available systems, unplanned failures are often the main point of discussion. However, if it takes 2 weeks to upgrade a system to a new revision of software, there are bound to be a large number of complaints.

The following sections discuss ways of handling the different types of planned downtime.

Reducing Time Needed for Application Upgrades and Patches

Once a year or so, a new revision of an application is released. How long does it take for the end-user to upgrade to this new revision? This answer is the amount of planned downtime a user must take to upgrade their application. The following guidelines reduce this time.

Provide for Rolling Upgrades

Provide for a "rolling upgrade" in a client/server environment. For a system with many components, the typical scenario is to bring down the entire system, upgrade every node to the new version of the software, and then restart the application on all the affected nodes. For large systems, this could result in a long downtime. An alternative is to provide for a rolling upgrade. A rolling upgrade rolls out the new software in a phased approach by upgrading only one component at a time. For

example, the database server is upgraded on Monday, causing a 15 minute downtime. Then on Tuesday, the application server on two of the nodes is upgraded, which leaves the application servers on the remaining nodes online and causes no downtime. On Wednesday, two more application servers are upgraded, and so on. With this approach, you avoid the problem where everything changes at once, plus you minimize long outages.

The trade-off is that the application software must operate with different revisions of the software. In the above example, the database server might be at revision 5.0 while the some of the application servers are at revision 4.0. The application must be designed to handle this type of situation.

Do Not Change the Data Layout Between Releases

Migration of the data to a new format can be very time intensive. It also almost guarantees that rolling upgrade will not be possible. For example, if a database is running on the first node, ideally, the second node could be upgraded to the new revision of the database. When that upgrade is completed, a brief downtime could be scheduled to move the database server from the first node to the newly upgraded second node. The database server would then be restarted, while the first node is idle and ready to be upgraded itself. However, if the new database revision requires a different database layout, the *old* data will not be readable by the newly updated database. The downtime will be longer as the data is migrated to the new layout.

Providing Online Application Reconfiguration

Most applications have some sort of configuration information that is read when the application is started. If to make a change to the configuration, the application must be halted and a new configuration file read, downtime is incurred.

To avoid this downtime use configuration tools that interact with an application and make dynamic changes online. The ideal solution is to have a configuration tool which interacts with the application. Changes are made online with little or no interruption to the end-user. This tool must be able to do everything online, such as expanding the size of the data, adding new users to the system, adding new users to the application, etc. Every task that an administrator needs to do to the application system can be made available online.

Documenting Maintenance Operations

Standard procedures are important. An application designer should make every effort to make tasks common for both the highly available environment and the normal environment. If an administrator is accustomed to bringing down the entire

system after a failure, he or she will continue to do so even if the application has been redesigned to handle a single failure. It is important that application documentation discuss alternatives with regards to high availability for typical maintenance operations.

D Integrating HA Applications with MC/ServiceGuard

The following is a summary of the steps you should follow to integrate an application into the MC/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 MC/ServiceGuard?
4. Identify problem areas
   - What does the application do today to handle a system reboot or panic?
   - Does the application use any system-specific information such as uname() or gethostname(), SPU_ID or MAC address which would prevent it from failing over to another system?

Checklist for Integrating HA Applications

This section contains a checklist for integrating HA applications in both single and multiple systems.

Defining Baseline Application Behavior on a Single System

1. Define a baseline behavior for the application on a standalone system:
   - Install the application, database, and other required resources on one of the systems. Be sure to follow MC/ServiceGuard rules in doing this:
     - Install all shared data on separate external volume groups.
     - Use JFS filesystems as appropriate.
   - Perform some sort of standard test to ensure the application is running correctly. This test can be used later in testing with MC/ServiceGuard. If possible, try to connect to the application through a client.
   - Crash the standalone system, reboot it, and test how the application starts up again. Note the following:
     - Are there any manual procedures? if so, document them.
     - Can everything start up from rc scripts?
   - Try to write a simple script which brings everything up without having to do any keyboard typing. Figure out what the administrator would do at the keyboard, then put that into the script.
   - Try to write a simple script to bring down the application. Again, figure out what the administrator would do at the keyboard, then put that into the script.

Integrating HA Applications in Multiple Systems

1. Install the application on a second system.
   - Create the LVM infrastructure on the second system.
   - Add the appropriate users to the system.
   - Install the appropriate executables.

- With the application *not* running on the first system, try to bring it up on the second system. You might use the script you created in the step above. Is there anything different that you must do? Does it run?
- Repeat this process until you can get the application to run on the second system.

2. Configure the MC/ServiceGuard cluster:
   - Create the cluster configuration.
   - Create a package.
   - Create the package script.
   - Use the simple scripts you created in earlier steps as the customer defined functions in the package control script.
3. Start the cluster and verify that applications run as planned.

Testing the Cluster

1. Test the cluster:
   - Have clients connect.
   - Provide a normal system load.
   - Halt the package on the first node and move it to the second node:

     ```
     # cmhaltpkg pkg1
     # cmrunpkg -n node2 pkg1
     # cmmodpkg -e pkg1
     ```

   - Move it back.

     ```
     # cmhaltpkg pkg1
     # cmrunpkg -n node1 pkg1
     # cmmodpkg -e pkg1
     ```

   - Fail one of the systems. For example, turn off the power on node 1. Make sure the package starts up on node 2.
   - Repeat failover from node 2 back to node 1.
2. Be sure to test all combinations of application load during the testing. Repeat the failover processes under different application states such as heavy user load versus no user load, batch jobs vs online transactions, etc.

3. Record timelines of the amount of time spent during the failover for each application state. A sample timeline might be 45 seconds to reconfigure the cluster, 15 seconds to run `fsck` on the filesystems, 30 seconds to start the application and 3 minutes to recover the database.

E Rolling Software Upgrades

You can upgrade the HP-UX operating system and the MC/ServiceGuard software one node at a time without bringing down your clusters. This process can also be used any time one system needs to be taken offline for hardware maintenance or patch installations. Until the process of upgrade is complete on all nodes, you cannot change the cluster configuration files, and you will not be able to use any of the features of the new MC/ServiceGuard release.

Refer also to the section on hardware maintenance in the "Troubleshooting" chapter.

Steps for Rolling Upgrades

Use the following steps:

1. Use `cmhaltnode` to halt the node you wish to upgrade. This will cause the node's packages to start up on an adoptive node.
2. Edit the `/etc/rc.config.d/cmcluster` file to include the following line:

   ```
   AUTOSTART_CMCLD = 0
   ```

3. Upgrade the node to the available HP-UX release, including MC/ServiceGuard. You can perform other software or hardware upgrades if you wish, provided you do not detach any SCSI cabling.
4. Edit the /etc/rc.config.d/cmcluster file to include the following line:

   ```
   AUTOSTART_CMCLD = 1
   ```

5. Restart the cluster on the upgraded node via the `cmrunnode` command.
6. Repeat this process for each node in the cluster.

NOTE Be sure to plan sufficient system capacity to allow moving the packages from node to node during the process without an unacceptable loss of performance.

If a cluster were to fail before the rolling upgrade was complete (perhaps due to a catastrophic power failure), the cluster can be restarted by entering the `cmruncl` command from a node which has been upgraded to the latest revision of the software.

Using SAM to Perform a Rolling Upgrade

A rolling upgrade can be carried out using SAM instead of HP-UX commands. However, SAM must be invoked 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.

To perform a rolling upgrade using SAM, from the SAM main menu, choose Clusters, then High Availability Clusters, and then select the appropriate type of administration — cluster or package administration. Perform the tasks as explained in the steps in the following section, using SAM instead of HP-UX commands.

Keeping Kernels Consistent

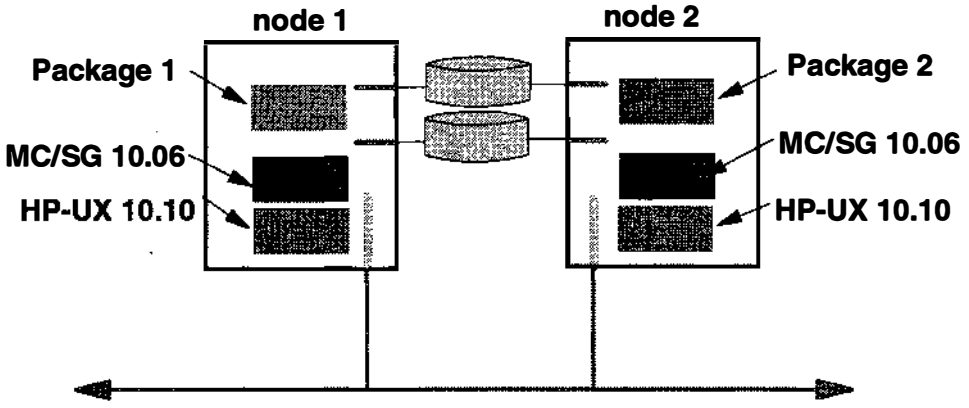
If you change kernel parameters as a part of doing a rolling upgrade, be sure to change the parameters similarly on all nodes that can run the same packages in a failover scenario.

Example of Rolling Upgrade

While you are performing a rolling upgrade warning messages may appear while the node is determining what version of software is running. This is a normal occurrence and not a cause for concern.

The following example shows a simple rolling upgrade on two nodes running one package each, as shown in Figure E-1. (This and the following figures show the starting point of the upgrade as MC/ServiceGuard 10.06 and HP-UX 10.10 for illustration only. A roll to MC/ServiceGuard 10.10 and HP-UX 10.20 is shown. For your systems, substitute the actual release numbers of your rolling upgrade path.)

Figure E-1 **Running Cluster Before Rolling Update**



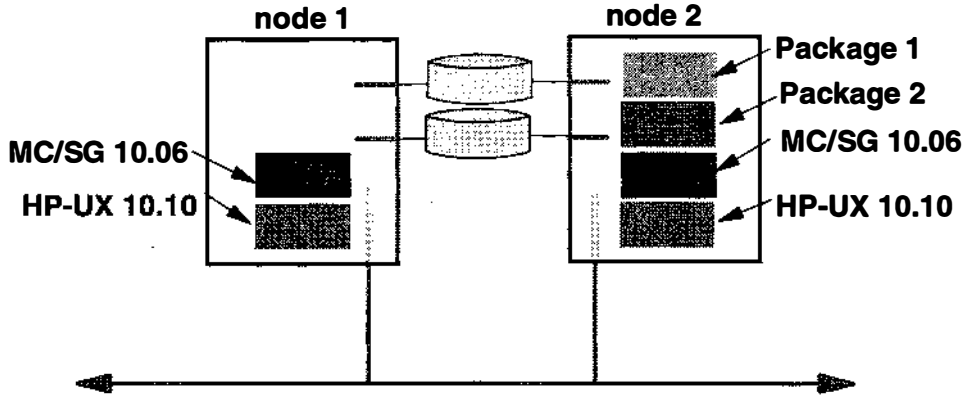
Step 1.

Halt the first node, as follows

```
# cmhaltnode -f node1
```

This will cause PKG1 to be halted cleanly and moved to node 2. The MC/ServiceGuard daemon on node 1 is halted, and the result is shown in Figure E-2.

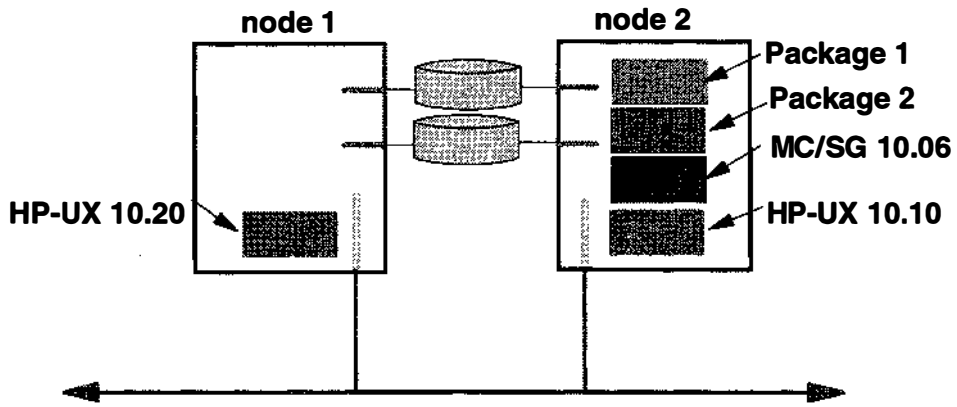
Figure E-2 **Running Cluster with Packages Moved to Node 2**



Step 2.

Upgrade node1 to the next operating system release (in this example, HP-UX 10.20), and install the next version of MC/ServiceGuard (10.10).

Figure E-3 **Node 1 Upgraded to HP-UX 10.20**



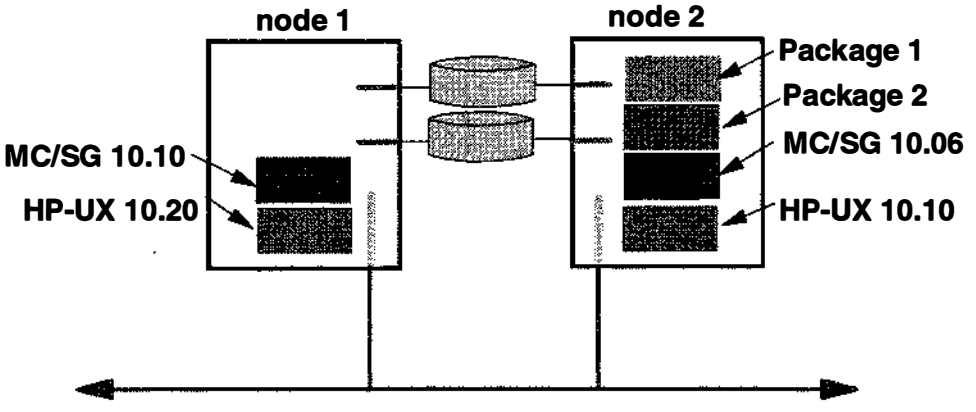
Step 3.

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 MC/ServiceGuard daemon (*cmcld*) are running on the two nodes, as shown in Figure E-4.

Figure E-4 **Node 1 Rejoining the Cluster**



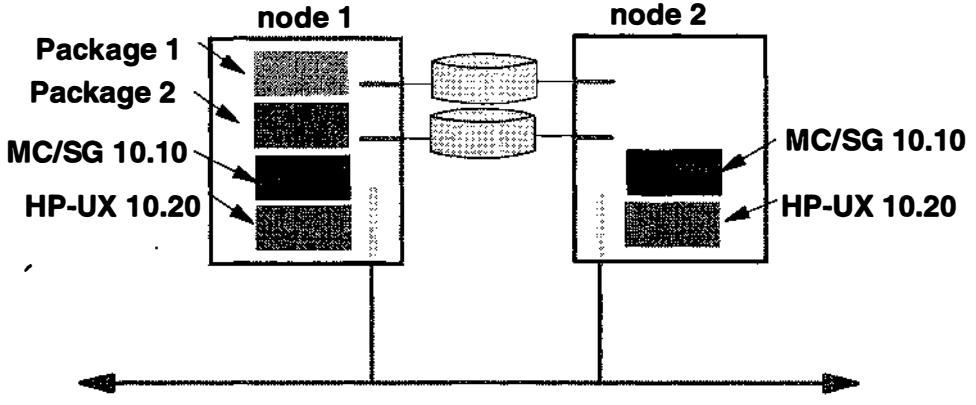
Step 4.

Repeat the process on node 2. Halt the node, as follows:

```
# cmhaltnode -f node2
```

This causes both packages to move to node 1. Then upgrade node 2 to HP-UX 10.20 and MC/ServiceGuard 10.10.

Figure E-5 **Running Cluster with Packages Moved to Node 1**



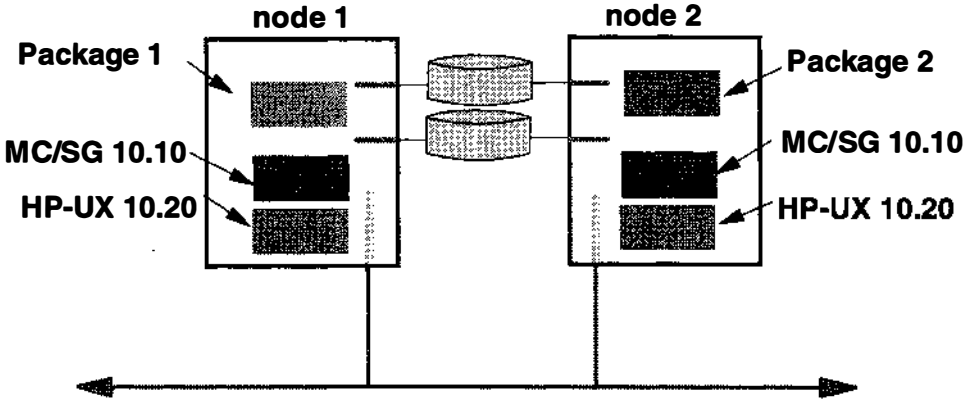
Step 5.

Move PKG2 back to its original node. Use the following commands:

```
# cmhaltpkg pkg2
# cmrunpkg -n node2 pkg2
# cmmodpkg -e pkg2
```

The `cmmodpkg` command re-enables switching of the package, which is disabled by the `cmhaltpkg` command. The final running cluster is shown in Figure E-6.

Figure E-6 **Running Cluster After Upgrades**



Limitations of Rolling Upgrades

The following limitations apply to rolling upgrades:

- During rolling upgrade, HP-UX commands and SAM interactive screens must be invoked on a node containing the latest revision of the software. Performing tasks on a node containing an earlier revision of the software will not work or will cause inconsistent results.
- You cannot modify the cluster or package configuration until the upgrade is complete. You *cannot* modfy the hardware configuration--including the cluster's network configuration--during rolling upgrade. This means that you must upgrade all nodes to the new release before you can modify the configuration file and copy it to all nodes.
- None of the features of the newer release of MC/ServiceGuard are allowed until all nodes have been upgraded.
- Binary configuration files may be incompatible between releases of MC/ServiceGuard. Do *not* manually copy configuration files between nodes.
- Rolling upgrades are only supported from the previous two releases. For more information, see the *MC/ServiceGuard Release Notes* for your particular release.
- You can perform a rolling upgrade only on a configuration that has not been modified since the last time the cluster was started.
- Rolling upgrades are not intended as a means of using mixed releases of MC/ServiceGuard or HP-UX within the cluster. It is highly recommended that you upgrade all cluster nodes as quickly as possible to the new release level.
- You cannot delete MC/ServiceGuard software (via **`swremove`**) from a node while the cluster is in the process of rolling upgrade.

F Using OTS/9000 on the LAN

MC/ServiceGuard supports OTS/9000, Hewlett-Packard's implementation of OSI Transport Services and related protocols. To configure OTS/9000 with MC/ServiceGuard, perform the following steps:

1. Configure OTS/9000 on all cluster nodes. See your OTS/9000 documentation for complete information.
2. Configure an IP subnet for each LAN card that has OTS on it. This is needed because MC/ServiceGuard actually monitors the LANICS (LAN interface cards) through configured IP subnets. See Chapter 3 for information on specifying IP subnets in MC/ServiceGuard.

When the cluster is running, MC/ServiceGuard will detect network failures by sensing an IP subnet failure.

NOTE In the MC/ServiceGuard environment, OTS/9000 protocol can only be used to transmit data, not heartbeats. However, you can configure a TCP/IP network for heartbeats on the same LAN that is running OTS/9000.

You can configure an OTS monitor as a package dependency within MC/ServiceGuard. Use the SAM interface, and choose "Additional Package Dependencies" in the "Package Configuration" subarea.

G Blank Planning Worksheets

This appendix reprints blank versions of the planning worksheets described in the "Planning" chapter. You can duplicate any of these worksheets that you find useful and fill them in as a part of the planning process.

Blank Hardware Worksheet

SPU Information:

S800 Host Name _____ S800 Series No _____

Memory Capacity _____ Number of I/O Slots _____

LAN Information:

Name of Subnet	Name of Interface	Node IP Addr	Traffic Type
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

Serial (RS232) Heartbeat Interface Information:

Node Name _____ RS232 Device File _____

Node Name _____ RS232 Device File _____

X.25 Information

OTS subnet _____ OTS subnet _____

Disk I/O Information for Shared Disks:

Bus Type _____ Slot Number ____ Address ____ Disk Device File _____

Bus Type _____ Slot Number ____ Address ____ Disk Device File _____

Bus Type _____ Slot Number ____ Address ____ Disk Device File _____

Bus Type _____ Slot Number ____ Address ____ Disk Device File _____

Attach a printout of the output from 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.

Blank Power Supply Worksheet

SPU Power:

S800 Host Name _____ Power Supply _____

S800 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 _____

Blank Volume Group and Physical Volume Worksheet

Volume Group Name: _____

Name of First Physical Volume Group: _____

Physical Volume Name: _____

Physical Volume Name: _____

Physical Volume Name: _____

Physical Volume Name: _____

Physical Volume Name: _____

Physical Volume Name: _____

Physical Volume Name: _____

Name of Second Physical Volume Group: _____

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: _____

Node Names: _____

Maximum Configured Packages: _____

Cluster Volume Groups: _____

Subnets:

Heartbeat IP Addresses: _____

Non-Heartbeat IP Addresses: _____

Cluster Lock Volume Groups and Volumes:

First Lock VG:     |    First Lock Physical Volume:
                   |
 _________________ |Name (Node 1): _____Disk Unit No: _____Power Unit:____
                   |
                   |Name (Node 2): _____Disk Unit No: _____Power Unit:____
                   |
Second Lock VG:    |  Second Lock Physical Volume:
                   |
 _________________ |Name (Node 1): _____Disk Unit No: _____Power Unit:____
                   |
                   |Name (Node 2): _____Disk Unit No: _____Power Unit:____

Timing Parameters:

Heartbeat Interval: _____

Node Timeout: _____

Network Polling Interval: _____

Autostart Delay: _____
```

Package Configuration Worksheet

```
Package Configuration File Data:

Package Name: _____

Primary Node: _____

First Failover Node:_____

Second Failover Node:_____

Package Run Script: _____Timeout: _____

Package Halt Script: _____Timeout: _____

Package Switching Enabled? _____ Local Switching Enabled? _____

Node Failfast Enabled? _____

Additional Package Resource:

Resource Name:_____ Polling Interval_____ Resource UP Value_____

Package Control Script Data:

VG[0]_____LV[0]_____FS[0]_____FS_MOUNT_OPT[0]_____

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

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

IP[0] _____ SUBNET[0]_____

IP[1] _____ SUBNET[1]_____

X.25 Resource Name _____

Service Name: _____ Run Command: _____Retries: _____

Service Fail Fast Enabled? _____Service Halt Timeout _____

Service Name: _____ Run Command: _____ Retries: _____

Service Fail Fast Enabled? _____Service Halt Timeout _____
```

Index

Index

Index

Index

Index

Index

Index

Index

Index

W

Index