HP-UX 11i Version 3
HP-UX System Administrator's Guide: Logical Volume Management
HP 9000 and HP Integrity Systems

HP Part Number: 5991-6481
Published: E0207
Edition: Edition 1

Legal Notices

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

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

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

U.S. Government License Proprietary computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Trademark Notices UNIX® is a registered trademark in the United States and other countries, licensed exclusively through The Open Group. VERITAS® is a registered trademark of Symantec Corporation.

Table of Contents

List of Figures

List of Tables

Preface

Intended Audience

The *HP-UX System Administrator's Guide* is written for administrators of HP-UX systems of all skill levels needing to administer HP-UX systems beginning with HP-UX Release 11i Version 3.

While many topics in this set apply to previous releases, much has changed in HP-UX 11i as of Version 3. Therefore, for information about prior releases please see the manual *Managing Systems and Workgroups: A Guide for System Administrators*.

About this Series

The *HP-UX System Administrator's Guide* documents the core set of tasks (and associated concepts) necessary to administer HP-UX 11i based systems as of HP-UX 11i Version 3.

The *HP-UX System Administrator's Guide* is a set of documents, comprised of the following volumes:

Overview	Provides a high-level view of HP-UX 11i, its components, and how they relate to each other.
Configuration Management	Describes many of the tasks you need to perform to configure and customize system settings and the behavior of subsystems.
Logical Volume Management	Documents how to configure physical volumes, volume groups, and logical volumes using the HP Logical Volume Manager (LVM).
Security Management	Documents the data and system security features of HP-UX 11i.
Routine Management Tasks	Documents many of the ongoing tasks you need to perform to keep your system running smoothly.

About this Document

HP-UX System Administrator's Guide: Logical Volume Management describes how to configure, administer, and troubleshoot the Logical Volume Manager (LVM) product on HP-UX platforms. It is divided into the following chapters, each of which contains information about configuring or troubleshooting LVM. This document also includes appendixes that contain supplemental information.

Chapter 1: Introduction	Use this chapter to learn about LVM.
Chapter 2: Configuring LVM	Use this chapter to learn how to configure a system to use LVM.

Chapter 3: Administering LVM	Use this chapter to learn how to administer LVM on an ongoing basis.
Chapter 4: Troubleshooting LVM	Use this chapter to learn how to troubleshoot problems in an LVM configuration.
Appendix A: LVM Specifications and Limitations	Use this appendix to find product specifications.
Appendix B: Cheatsheet	Use this appendix to find frequently used LVM commands.
Appendix C: Glossary	Use this appendix to find definitions of frequently used LVM terms.

Related Information

Additional information about LVM and HP-UX can be found at http://docs.hp.com in the *HP-UX Operating Environment* collection. In particular, the following white papers are available:

- *LVM Limits*
- *LVM Online Disk Replacement (LVM OLR)*
- *SLVM Single-Node Online Reconfiguration (SLVM SNOR)*
- *When Good Disks Go Bad: Dealing with Disk Failures under LVM*

Finding HP-UX Information

The following table outlines where to find general system administration information for HP-UX. This table does not include information for specific products.

Table 1 Finding HP-UX Information

If you need to. . .	Refer To...	Located at . . .
Find out: • what has changed in HP-UX releases • The content of the Operating Environments • Firmware requirements, and supported systems for a specific release	The HP-UX 11i Release Notes specific to your version of HP-UX.	• HP Instant Information media • HP-UX Technical Documentation web site **http://docs.hp.com**
Install or update HP-UX	• *Read Before Installing or Updating to HP-UX* • *HP-UX 11i Installation and Update Guide*	• Media Kit (supplied with the Operating Environment) • HP Instant Information media • HP-UX Technical Documentation web site **http://docs.hp.com**
Administer an HP-UX system	For Releases Prior to HP-UX 11i Version 3: • *Managing Systems and Workgroups: A Guide for HP-UX System Administrators* For Releases beginning with HP-UX 11i Version 3: • *HP-UX System Administrator's Guide* (a multi-volume set) Other sources of System Administration Information: • *HP System Partitions Guide: Administration for nPartitions* • "Planning Superdome Configurations" (white paper)	• HP Instant Information media • HP-UX Technical Documentation web site **http://docs.hp.com** • "Planning Superdome Configurations" is available at http://docs.hp.com/hpux/onlinedocs/os/11i/upcd.pdf

HP-UX 11i Release Names and Operating System Version Identifiers

With HP-UX 11i, HP delivers a highly available, secure, and manageable operating system that meets the demands of end-to-end Internet-critical computing. HP-UX 11i supports enterprise, mission-critical, and technical computing environments. HP-UX 11i is available on both HP 9000 systems and Integrity systems.

Each HP-UX 11i release has an associated release name and release identifier. The `uname` command with the `-r` option returns the release identifier. Table 2 "HP-UX 11i Releases" shows the releases available for HP-UX 11i.

Table 2 HP-UX 11i Releases

OS Version Identifier	Release Name	Supported Processor Architecture
B.11.11	HP-UX 11i Version 1	HP 9000
B.11.23	HP-UX 11i Version 2	Integrity
B.11.23.0409	HP-UX 11i Version 2 September 2004 Update	HP 9000 and Integrity
B.11.31	HP-UX 11i Version 3	HP 9000 and Integrity

For information on supported systems and processor architecture for various versions of HP-UX 11i, refer to the HP-UX 11i system release notes specific to your version of HP-UX (for example, the *HP-UX 11i Version 3 Release Notes*).

Determining Your System Version

The `uname`, `model`, and `swlist` commands can help you determine information about your system, including its hardware type, machine model, operating system version, and operating environment update status. (See *uname*(1), *model*(1), and *swlist*(1M).)

For OS naming conventions, please see "HP-UX 11i Release Names and Operating System Version Identifiers" (page 15).

Table 3 OS Version, System Architecture, and Machine Model

Topic	Command	Sample Output
OS Version	`$ uname -r`	`B.11.31`[1]
Architecture	`$ uname -m`	`ia64`[2] `9000/800`[2]
Machine Model	`$ model`[3]	`ia64 hp server rx5670` `9000/800/S16K-A`
Operating Environment	`$ swlist HPUX*OE*`	`# HPUX11i-OE-MC B.11.31 HP-UX Mission Critical Operating Environment`[1]
OS Version.Update	`$ swlist HPUX*OE*`	`# HPUX11i-TCOE B.11.23.0409 HP-UX Technical Computing OE Component`[1]

1 HP-UX 11i OS version identifiers have the form B.11.23 or B.11.23.0409, where B.11.23 is the OS version and 0409 is the year-month of the operating environment (OE) update.

2 `ia64` = Integrity. All others = HP 9000.

3 The `getconf MACHINE_MODEL` command gives the same output (see *getconf*(1)).

Typographic Conventions

This document uses the following typographical conventions.

audit(5)	An HP-UX manpage. *audit* is the name and *5* is the section in the *HP-UX Reference*. On the web and on the Instant Information media, it may be a hot link to the manpage itself. From the HP-UX command line, you can enter `man audit` or `man 1 audit` to view the manpage. See *man*(1) for more information.
Document Title	The title of a document. On the web and on the Instant Information media, it may be a hot link to the document itself.
`Command`	A command name or qualified command phrase.
`ComputerOut`	Text displayed by the computer.
Emphasis	Text that is emphasized.
Emphasis	Text that is strongly emphasized.
KeyCap	The name of a keyboard key. Note that **Return** and **Enter** both refer to the same key.
Term	The defined use of an important word or phrase.
`UserInput`	Commands and other text that the user types.
`Variable`	The name of a variable that you may replace in a command or function or information in a display that represents several possible values.
`$`	User command prompt.
`#`	Superuser (`root`) command prompt.

Examples and Shells

This document describes practices used by the system administrator. Since the `root` user, a superuser, is required to use the POSIX shell `/sbin/sh`, all command examples use that shell. The POSIX shell is defined in *sh-posix*(1). For information on other shells, see the *Shells User's Guide* and *sh*(1).

Command Syntax

`Literal`	A word or character that you enter literally.
`Replaceable`	A word or phrase that you replace with an appropriate value.
-chars	One or more grouped command options, such as `-ikx`. The *chars* are usually a string of literal characters that each represent a specific option. For example, the entry `-ikx` is equivalent to the individual

	options `-i`, `-k`, and `-x`. The plus character (+) is sometimes used as an option prefix.
-word	A single command option, such as `-help`. The *word* is a literal keyword. The difference from *-chars* is usually obvious and is clarified in an Options description. The plus character (+) and the double hyphen (`--`) are sometimes used as option prefixes.
[ ]	The bracket metacharacters enclose optional content in formats and command descriptions.
{ }	The brace metacharacters enclose required content in formats and command descriptions.
|	The bar metacharacter separates alternatives in a list of choices, usually in brackets or braces.
. . .	The ellipsis metacharacter after a token (`abc . . .`) or a right bracket (`[ ] . . .`) or a right brace (`{ } . . .`) metacharacter indicates that the preceding element and its preceding whitespace, if any, may be repeated an arbitrary number of times.
. . .	Ellipsis is sometimes used to indicate omitted items in a range.

Publication History

The document publication date and part number indicate its current edition. The publication date will change when a new edition is released.

To ensure that you receive the new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition	February 2007
	HP Part Number 5991-6481
	HP-UX 11i Version 3
	Print, Instant Information DVD, and HP Technical Documentation web site (http://docs.hp.com)

NOTE: The volumes in the *HP-UX System Administrator's Guide* may be updated independently. Therefore, the latest versions of the volumes in the set may vary with time, with respect to each other. The latest versions of each volume are available at http://docs.hp.com.

HP Encourages Your Comments

HP encourages your comments concerning this document. We are truly committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments through our technical documentation feedback website:

http://docs.hp.com/en/feedback.html

Include document title, manufacturing part number, and any comment, error found, or suggestion for improvement you have concerning this document. Also, include what we did right so we can incorporate it into other documents.

1 Introduction

This chapter contains information on the following topics:

LVM Features

Logical Volume Manager (LVM) is a storage management system that lets you allocate and manage disk space for file systems or raw data. Historically, you would treat your disks individually and in terms of fixed-sized partitions; each disk or partition would hold a file system, swap space, boot area, or raw data. With LVM, you do not have to assign a disk or fixed-sized partition to a single purpose. Instead, you consider the disks as a pool (or volume) of data storage, consisting of equally sized extents. Extents are allocated into virtual storage devices known as **logical volume**s, which can be treated as disks.

LVM provides capabilities that are unavailable when using only fixed partitions:

- A logical volume's size can be dynamically reduced or expanded to meet changing data needs. For example, a logical volume can be as small or large as the file system mounted to it requires. The file system can be extended without rebuilding it; reducing a file system is more complex, and may require rebuilding it.
- Small chunks of unused space from several disks can be combined to create a usable volume.
- A logical volume can exceed the size of a physical disk. This feature is known as **disk spanning**, because a single file system (and individual files) can span disks.
- Up to three copies of identical data can be stored and updated simultaneously using LVM. This feature is known as **mirroring** a logical volume, and requires an optional product, HP MirrorDisk/UX. See "Increasing Data Redundancy Through Mirroring" (page 38).
- Mirrored data can be configured to automatically create a new mirror to a separate disk when one of the mirror copies fails. This feature is known as **sparing**, and requires an optional product, HP MirrorDisk/UX. See "Increasing Hardware Redundancy Through Disk Sparing" (page 44).
- A logical volume can be created such that logically contiguous data blocks (for example, chunks of the same file) are distributed across multiple disks, which speeds I/O throughput for large files when they are read and written sequentially.

This feature is known as **striping**. Striping can be used in conjunction with mirroring. See "Increasing Performance Through Disk Striping" (page 46).

- Devices that are accessible through multiple links can be configured to improve availability. If the primary link to a device fails, LVM can switch automatically to an alternate link. This feature is known as **multipathing**. See "Increasing Availability Through Multipathing" (page 50).

LVM Architecture

An LVM system starts with initializing disks for LVM usage. An LVM disk is known as a **physical volume**, or PV. A disk is marked as an LVM physical volume using either the System Management Homepage (HP SMH) or the `pvcreate` command. Physical volumes use the same device special files as traditional HP-UX disk devices.

LVM divides each physical volume into addressable units called **physical extents**. Starting after the LVM metadata at the beginning of the disk, extents are allocated sequentially, with index zero and incrementing the index by one for each unit. The physical extent size is configurable at the time you form a volume group and applies to all disks in the volume group. By default, each physical extent has a size of 4 MB. This value can be set when you create the volume group to a value between 1 MB and 256 MB.

Physical volumes are organized into volume groups. A **volume group** can consist of one or more physical volumes, and there can be more than one volume group in the system. Once created, the volume group, and not the disk, is the entity that represents data storage. Thus, whereas earlier one would move disks from one system to another, with LVM, one would move a volume group from one system to another. For this reason it is often convenient to have multiple volume groups on a system.

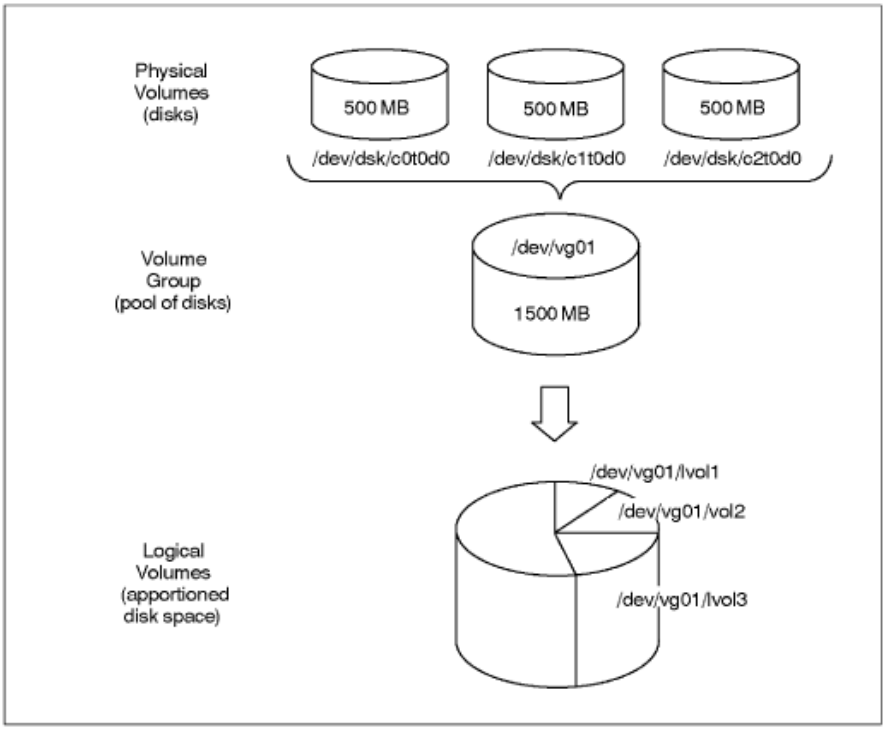
The pool of disk space that is represented by a volume group can be apportioned into **logical volume**s of various sizes. Once created, logical volumes can be treated just like disk partitions. They are accessible through device special files. A logical volume can span a number of physical volumes in a volume group or represent only a portion of one physical volume.

The basic allocation units for a logical volume are called **logical extents**. A logical extent is mapped to a physical extent; thus, if the physical extent size is 4 MB, the logical extent size will also be 4 MB. The size of a logical volume is determined by the number of logical extents configured.

You then assign file systems, swap, dump, or raw data to logical volumes. For example, in Figure 1-1, logical volume `/dev/vg01/lvol1` might contain a file system, logical volume `/dev/vg01/lvol2` might contain swap space, and logical volume `/dev/vg01/lvol3` might contain raw data. You can use HP SMH to create a file system in a logical volume of a specified size, and then mount the file system, or you can use LVM commands to create and then extend a logical volume to allocate sufficient

space for file system, or raw data. You would then create and mount new file systems or install your application in the logical volume.

Figure 1-1 Disk Space Partitioned Into Logical Volumes



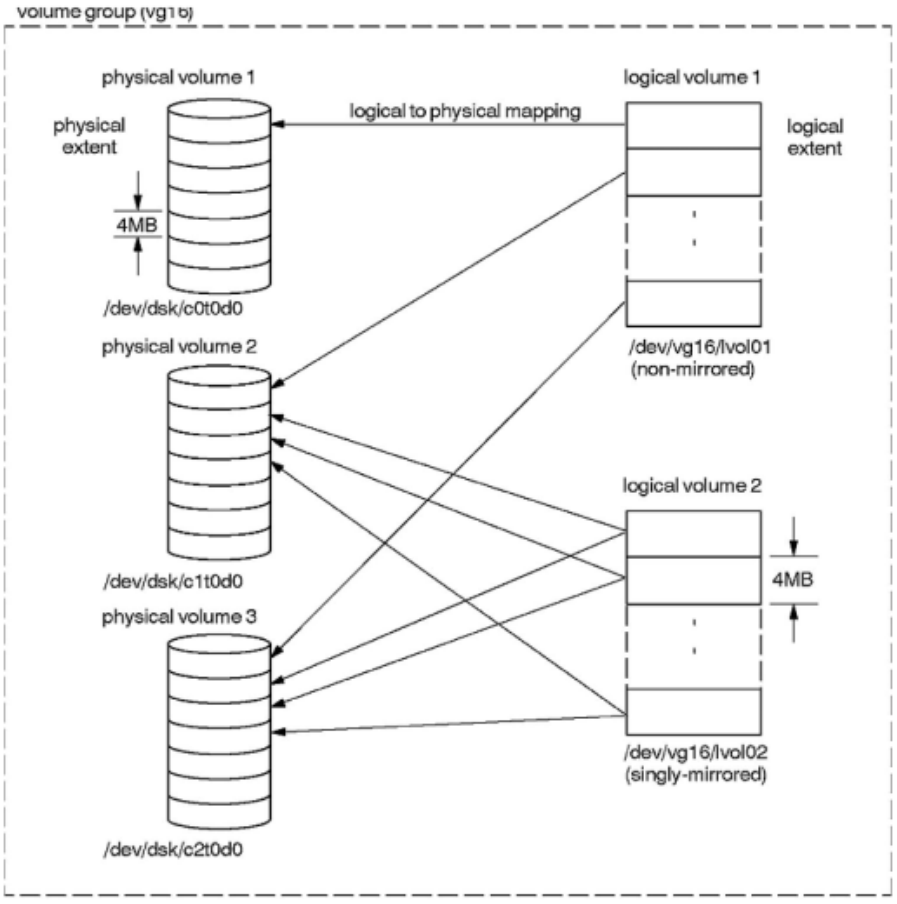
Physical versus Logical Extents

When LVM allocates disk space to a logical volume, it automatically creates a mapping of the logical extents to physical extents. This mapping depends on the policy chosen when creating the logical volume. Logical extents are allocated sequentially, starting at zero, for each logical volume. LVM will use this mapping to access the data, regardless of where it physically resides. Commands are provided for you to examine this mapping; see *pvdisplay*(1M) and *lvdisplay*(1M).

Except for mirrored, striped, or striped-mirrored logical volumes, each logical extent is mapped to one physical extent. For mirrored logical volumes, each logical extent is mapped to either two or three physical extents, depending on whether you are using 1–way or 2–way mirroring. For example, if one mirror copy exists, then each logical extent maps to two physical extents, one extent for the original and one for the mirror copy. See "Increasing Data Redundancy Through Mirroring" (page 38) for more information on mirroring. For information on striped logical volumes, see "Increasing Performance Through Disk Striping" (page 46). Also, refer to the book *Disk and File Management Tasks on HP-UX*.

Figure 1-2 shows an example of several types of mapping available between physical extents and logical extents within a volume group.

Figure 1-2 Physical Extents and Logical Extents



As shown in Figure 1-2, the contents of the first logical volume are contained on all three physical volumes in the volume group. Because the second logical volume is mirrored, each logical extent is mapped to more than one physical extent. In this case, there are two physical extents containing the data, each on both the second and third disks within the volume group.

By default, LVM assigns physical extents to logical volumes by selecting available physical extents from disks in the order in which they appear in the LVM configuration file, `/etc/lvmtab`. As a system administrator, you can bypass this default assignment and control which disks are used by a logical volume (see "Extending a Logical Volume to a Specific Disk" (page 68)).

If a logical volume is to be used for root, boot, primary swap, or dump, the physical extents must be **contiguous**, which means that the physical extents must be allocated in increasing order with no gaps on a single physical volume. For logical volumes that are not being used for root, boot, primary swap or dump, physical extents that correspond to contiguous logical extents within a logical volume can be **noncontiguous** on a physical volume or reside on entirely different disks. As a result, a file system created within one logical volume can reside on more than one disk.

LVM Device File Usage

All LVM components are represented by device files located in the `/dev` directory. (You might think of device files as agents for managing the interactions with the disk space.) The LVM device files are created by both HP SMH and HP-UX commands. This section describes LVM's use of device special files, and naming conventions for LVM objects.

Legacy Device Files versus Persistent Device Files

As of HP-UX 11i Version 3, disk devices can be represented by two different types of device files in the `/dev` directory: **legacy** and **persistent**.

Legacy device files were the only type of mass storage device files in releases prior to HP-UX 11i Version 3. They have hardware path information such as SCSI bus, target, and LUN encoded in the device file name and minor number. For example, the legacy device file `/dev/dsk/c3t2d0` represents the disk at card instance 3, target address 2, and lun address 0.

Persistent device files are not tied to the physical hardware path to a disk, but instead map to the disk's unique world-wide identifier (WWID). Thus, the device file is unchanged if the disk is moved from one interface to another, moved from one switch/hub port to another, or presented via a different target port to the host. The name of a persistent device file follows a simpler naming convention: `/dev/disk/disk`*n*, where *n* is the instance number assigned to the disk. Neither the device file name nor the minor number contain any hardware path information.

In addition, if the disk has multiple hardware paths, it is represented by a single persistent device file. Persistent device file transparently handle multipathed disks, and supersede LVM's multipathing functionality described in "Increasing Availability Through Multipathing" (page 50). If a disk has multiple hardware paths, which LVM refers to as **pvlinks**, the persistent device special file serves as a single access point for all the links. I/O requests are distributed across all available links by the mass storage stack, with a choice of load balancing algorithms. If a link fails, the mass storage stack automatically disables the failed link and I/O continues on all remaining links. Any failed or non-responsive links are monitored, so that when a failed link recovers, it is automatically and transparently reincorporated into any load balancing. New disks and links are also automatically discovered and added to load balancing. If there are any changes to the disk's connectivity—addition, removal, or modification of a

link—applications using the persistent device file are not affected, provided at least one link is still active. New disks also automatically discovered.

You can use either persistent or legacy device files for LVM disks. LVM recommends the use of persistent device special files, because they support a greater variety of load balancing options.

NOTE: If you want to use LVM's alternate link functionality, you must use legacy device files, and disable multipathing through those legacy device files, as described in "Increasing Availability Through Multipathing" (page 50).

Naming Conventions for LVM

You must refer to LVM devices or volume groups by name when using them within HP SMH or with HP-UX commands. By default, the LVM device files created by both HP SMH and HP-UX commands follow a standard naming convention. However, you can choose customized names for volume groups and logical volumes.

Naming Physical Volumes

Physical volumes are identified by their device file names, for example:

`/dev/disk/diskn`	Persistent block device file
`/dev/disk/diskn_p2`	Persistent block device file, partition 2
`/dev/rdisk/diskn`	Persistent character device file
`/dev/rdisk/diskn_p2`	Persistent character device file, partition 2
`/dev/dsk/cntndn`	Legacy block device file
`/dev/dsk/cntndns2`	Legacy block device file, partition 2
`/dev/rdsk/cntndn`	Legacy character device file
`/dev/rdsk/cntndns2`	Legacy character device file, partition 2

Each disk has a block device file and a character or raw device file, the latter identified by the `r`. Which name you use depends on what task you are doing with the disk.

For the boot disk on HP Integrity servers, make sure to use the device file with the `_p2` suffix or `s2` suffix, as that represents the HP-UX partition on the boot disk. On HP 9000 servers, use the device file without a partition number.

Use a physical volume's raw device file for these tasks only:

- When preparing a physical volume for LVM using the `pvcreate` command. Here, you use the device file for the disk. For example, this might be

/dev/rdisk/disk14. (The absence of a partition suffix indicates you are referring to the entire disk.)

- When removing LVM information from a physical volume using the pvremove command.
- When restoring your volume group configuration using the vgcfgrestore command.
- When performing a consistency check on a physical volume using the pvck command.
- When modifying the volume group identifier on a physical volume using the vgchgid command.

For all other tasks, use the block device file. For example, when you add a physical volume to a volume group using the vgextend command, you use the disk's block device file for the disk, such as /dev/disk/disk14.

All disk device files are created automatically when a new disk is discovered. Refer to *insf*(1M) for more information.

Naming Volume Groups

Each volume group must have a unique name, up to 255 characters. For example, typical volume group names could be vg01, vgroot, or vg_sales. Although the name does not have to start with vg, this is highly encouraged. By default, HP SMH will use the names of the form /dev/vg*nn*. The number *nn* starts at 00 and is incremented in the order that volume groups are created. By default, your root volume group will be vg00 although this name is not required.

Naming Logical Volumes

Logical volumes are identified by their device file names which can either be assigned by you or assigned by default when you create a logical volume using the lvcreate command.

When assigned by you, you can choose whatever name you want up to 255 characters.

When assigned by default, these names take the form: /dev/vg*nn*/lvol*N* (the block device file form) and /dev/vg*nn*/rlvol*N* (the character device file form). The number *N* starts at 1 and is incremented in the order that logical volumes are created within each volume group.

When LVM creates a logical volume, it creates both block and character device files. LVM then places the device files for a logical volume in the appropriate volume group directory.

For example, the default block name for the first logical volume created in volume group vg01 would have the full path name:

/dev/vg01/lvol1

If you create a logical volume to contain a sales database, you might want to name it explicitly:

```
/dev/vg01/sales_db_lv
```

After the logical volume in the above example has been created, it will have two device files: `/dev/vg01/sales_db_lv` for the block device file and `/dev/vg01/rsales_db_lv` for the character, or raw, device file.

Naming Physical Volume Groups

Physical volume groups are useful for mirroring and are discussed under "Increasing Performance Through I/O Channel Separation" (page 50). The only naming restriction in this case is that within a volume group, each physical volume group must have its own unique name. For example, the volume group `/dev/vg02` might have two physical volume groups called `/dev/vg02/pvg1` and `/dev/vg02/pvg2`.

LVM Command Summary

This section discusses LVM commands and their intended uses.

Managing LVM Using HP System Management Homepage

The HP System Management Homepage (HP SMH) enables you to perform most, but not all, LVM management tasks. Tasks that can be performed with HP SMH include:

- Creating or removing volume groups
- Adding or removing disks within volume groups
- Activating and deactivating volume groups
- Exporting and importing volume groups
- Creating, removing, or modifying logical volumes
- Increasing the size of logical volumes
- Creating or increasing the size of a file system in a logical volume
- Creating striped logical volumes
- Creating and modifying mirrored logical volumes
- Splitting a mirrored logical volume and merging a mirror copy
- Adding and removing mirror copies within mirrored logical volumes
- Creating and modifying physical volume groups

These tasks can also be performed with HP-UX commands. See "Administering LVM" (page 59) for more information.

To use HP SMH, enter the command `/usr/sbin/smh`.

For help using HP SMH, consult the HP SMH online help.

Managing LVM Using HP-UX Commands

As stated previously, all LVM management tasks can be performed using HP-UX commands.

The following tables give you general information on the commands you must use to perform a given task. Refer to the *HP-UX manpages* for detailed information.

Table 1-1 Commands Needed for Physical Volume Management Tasks

Task	Commands Needed
Changing the characteristics of a physical volume in a volume group	`pvchange`
Creating a physical volume for use in a volume group	`pvcreate`
Displaying information about physical volumes in a volume group	`pvdisplay`
Moving data from one physical volume to another	`pvmove`
Removing a physical volume from LVM control	`pvremove`
Checking or repairing a physical volume	`pvck`
Checking if a disk volume is under LVM control	`lvmchk`

Table 1-2 Commands Needed for Volume Group Management Tasks

Task	Commands Needed
Creating a volume group	`vgcreate` [1] [2]
Removing a volume group	`vgremove` [3]
Activating, deactivating, or changing the characteristics of a volume group	`vgchange`
Modifying the configuration parameters of a volume group; handling physical volume size changes	`vgmodify`
Backing up volume group configuration information	`vgcfgbackup` [4]
Restoring volume group configuration from a configuration file	`vgcfgrestore`
Displaying information about volume groups	`vgdisplay`
Exporting a volume group and its associated logical volumes	`vgexport`
Importing a volume group onto the system; adding an existing volume group back into `/etc/lvmtab`	`vgimport` [5]
Scanning all physical volumes looking for logical volumes and volume groups; recovering the LVM configuration file, `/etc/lvmtab`	`vgscan`
Adding a disk to a volume group	`vgextend` [6]
Removing a disk from a volume group	`vgreduce`

Table 1-2 Commands Needed for Volume Group Management Tasks *(continued)*

Task	Commands Needed
Synchronizing mirrored logical volumes in a volume group	`vgsync`[7]
Modifying the Volume Group ID on a physical volume	`vgchgid`
Migrating a volume group from legacy to persistent device files	`vgdsf`

1 Before executing command, one or more physical volumes must have been created with `pvcreate`.

2 You must also create a directory for the volume group and a `group` device file in the directory. See "Creating a Volume Group" (page 64) or *lvm*(7) for more information.

3 If logical volumes exist within the volume group, they must first be removed using the `lvremove` command. Also, the volume group must not contain more than one physical volume. If it does, use `vgreduce` first.

4 Invoked automatically whenever a configuration change is made by one of the following commands: `lvchange`, `lvcreate`, `lvextend`, `lvlnboot`, `lvmerge`, `lvreduce`, `lvrmboot`, `lvsplit`, `pvchange`, `vgcreate`, `vgextend`, `vgmodify`, or `vgreduce`.

5 You must also create a directory for the volume group and a `group` device file in the directory. See "Creating a Volume Group" (page 64) or *lvm*(7) for more information.

6 Before executing command, one or more physical volumes must have been initialized for LVM use with `pvcreate`.

7 Requires optional HP MirrorDisk/UX software.

Table 1-3 Commands Needed for Logical Volume Management Tasks

Task	Commands Needed
Creating a logical volume	`lvcreate`
Modifying a logical volume	`lvchange`
Displaying information about logical volumes	`lvdisplay`
Increasing the size of logical volume by allocating disk space	`lvextend`
Decreasing the size of a logical volume	`lvreduce`
Removing the allocation of disk space for one or more logical volumes within a volume group	`lvremove`
Preparing a logical volume to be a root, primary swap, or dump volume; updating the boot information on the boot physical volume	`lvlnboot` [1]
Removing link that makes a logical volume a root, primary swap, or dump volume	`lvrmboot`
Splitting a mirrored logical volume into two logical volumes	`lvsplit`[2]
Merging two logical volumes into one mirrored logical volume	`lvmerge`[3]
Synchronizing mirror copies in a mirrored logical volume	`lvsync`[4]

1 Invoked automatically whenever the configuration of the root volume group is affected by one of the following commands: `lvextend`, `lvmerge`, `lvreduce`, `lvremove`, `lvsplit`, `pvmove`, `vgextend`, or `vgreduce`.
2 Requires optional HP MirrorDisk/UX software.
3 Requires optional HP MirrorDisk/UX software.
4 Requires optional HP MirrorDisk/UX software.

LVM Limitations

LVM is a sophisticated subsystem; as such, it takes time to learn, it requires maintenance, and in rare cases, things can go wrong.

HP recommends using logical volumes as the preferred method for managing disks. You should certainly use LVM on file and application servers. On servers that have only a single disk and are used only to store the operating system and for swap, a "whole-disk" approach is simpler and easier to manage. LVM is not necessary on such systems, though you might choose to implement it for the sake of uniformity or because you expect to add more disks over time.

Your LVM configuration is, by default, automatically backed up every time you change it (in `/etc/lvmconf`). Mirroring provides insurance against data loss that is not available under the "whole-disk" method.

- Both LVM disks and non-LVM disks can exist simultaneously on your system, but a given disk (or partition) must be managed entirely by either LVM or non-LVM methods. That is, you cannot combine these techniques for use with a single disk (or partition).
- On an HP Integrity server, LVM supports partitioning of the root disk (and its mirrors) only, and supports only one HPUX partition on any disk.
- Although hard disk drives and disk arrays support the use of logical volumes, floppy disks, optical disks, and CD-ROMs do not.
- You must use an LVM or VERITAS Volume Manager (VxVM) disk for your root disk.
- To use LVM, a disk must be first initialized into a **physical volume**.
- To be allocatable for storage, a physical volume must be assigned to a volume group. If you think of all physical volumes as forming a storage pool, then a subset of disks from the pool can be joined together into a volume group.
- A given physical volume can only belong to one volume group.
- A volume group can contain from one to 255 physical volumes.
- A volume group can contain up to 255 logical volumes.
- A logical volume can exist on only one disk or reside on portions of many disks.
- The disk space within a logical volume can be used for swap, dump, raw data, or a file system.

- If a logical volume spans multiple physical volumes, it is not required that each disk be of the same interface type; however, having the same interface type will result in better performance.
- The extent size of a volume group is fixed when the volume group is created. It cannot be changed without recreating the volume group.

2 Configuring LVM

By default, the LVM commands are already installed on your system. This chapter covers what to consider when setting up your logical volumes. It contains information on the following topics:

Planning Your LVM Configuration

Using logical volumes requires some planning. Some of the issues you should consider for planning purposes are listed below and discussed in this chapter. You should consider these issues before setting up or modifying logical volumes on your system.

- For what purpose will you use a logical volume? For raw data or a file system? As a swap area or dump area? See "Setting Up Different Types of Logical Volumes" (page 34).
- How big should you make a logical volume?
- Does your data require high availability? If so, consider mirroring your logical volume across multiple disks, as described in "Increasing Data Redundancy Through Mirroring" (page 38). Also consider setting up spare disks to handle mirror failure, as described in "Increasing Hardware Redundancy Through Disk Sparing" (page 44).
- Is I/O performance very important to you? If so, consider striping your logical volumes across multiple disks, as described in "Increasing Performance Through Disk Striping" (page 46), and separating I/O channels, as described in "Increasing Performance Through I/O Channel Separation" (page 50). For additional recommendations for performance, read "Configuring for Performance" (page 55).
- Do you need to balance high availability and I/O performance? If so, consider striping and mirroring your logical volume, as described in "Increasing Data Redundancy Through Mirroring" (page 38) and "Increasing Performance Through Disk Striping" (page 46).
- Is it important to you to recover from disk failures quickly? If so, read "Configuring for Optimal Recovery" (page 52).

Setting Up Different Types of Logical Volumes

This section contains information on setting up special logical volumes.

Setting Up Logical Volumes for Raw Data Storage

You can optimize raw I/O performance by planning your logical volumes specifically for raw data storage. To create a raw data logical volume (such as for a database), consider how large to create the logical volume and how such a logical volume is distributed over your disks.

Typically, you specify the size of a logical volume in megabytes. However, a logical volume must be a multiple of the extent size used in the volume group. By default, the size of each logical extent is 4 MB.

For example, if a database partition requires 2002 MB and the default logical extent size is 4 MB, LVM creates a logical volume that is 2004 MB (or 501 logical extents).

If you plan to use logical volumes heavily for raw data storage (such as for setting up database partitions), consider how the logical volumes are distributed over your disks.

By default, LVM assigns disk space for a logical volume from one physical volume, uses the space on this physical volume entirely, and then assigns space from each successive physical volume in the same manner. LVM uses the physical volumes in the order in which they appear in `/etc/lvmtab`, which means that data of a logical volume might not be evenly distributed over all the physical volumes within your volume group.

As a result, when I/O access to the logical volumes occurs, one or more disks within the volume group might be heavily used, while the others might be lightly used, or not even used at all. This arrangement does not provide optimum I/O performance.

As a better alternative, you can set up your logical volume on specific disks in an interleaved manner, thus balancing the I/O access and optimizing performance (see "Extending a Logical Volume" (page 67)).

Because there are no HP-UX commands that will identify that the contents of a logical volume are being used for raw data, it is a good idea to name the logical volumes you create for raw data with easily recognizable names. In this way, you can recognize the contents of such a logical volume.

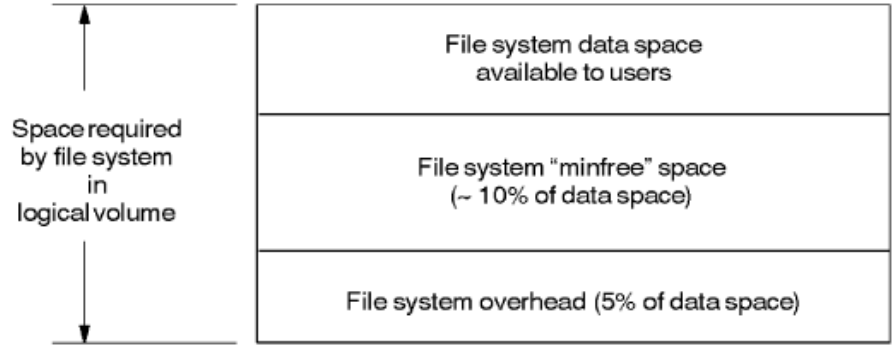
Setting Up Logical Volumes for File Systems

File systems reside in a logical volume just as they do within disk partitions or non-partitioned disks. Two types of file systems can be used in a logical volume: Hierarchical File Systems (HFS) and Journaled File Systems (JFS) (VxFS).

Choosing Initial Size of File System Logical Volumes

When determining the required space for a file system, consider the three major components shown in Figure 2-1.

Figure 2-1 File System Space Components



To get a rough estimate of how big to make a logical volume that will contain your file system:

1. Estimate how much disk space users will need for their data in the future. Allow for any anticipated changes, which usually include additional growth. (Use the `du` command to see how much disk space is currently being used.)
2. Add 10% to the above amount for a "`minfree`" area; this area is reserved to maintain performance.
3. Add another 5% for file system overhead; this includes all data structures required to maintain the file system.
4. Round up to the next integer multiple of the logical extent size used in this logical volume to find the size in logical extents. (Unlike the previous steps, this step is performed automatically for you when you create a logical volume.)

For example, suppose a group of users will require 60 MB space for file system data; this estimate allows for expected growth. Add 6 MB for the "`minfree`" space and arrive at 66 MB. Then add another 3 MB for file system overhead and arrive at a grand total estimate of 69 MB required by the file system, and by consequence, for the logical volume that contains the file system. If you are creating the logical volume in a volume group that has an extent size of 4 MB, round 69 to 72 to make it divisible by 4 MB. That is, LVM will create your logical volumes in multiples of the logical extent size.

Although these estimates are not precise, they suffice for planning how big to make a file system. You want your file system to be large enough for some useful time before having to increase its size.

TIP: Because increasing the size of a file system is usually much easier than reducing its size, you might benefit by being conservative in estimating how large to create a file system.

However, an exception to this consideration would be the root file system because, as a contiguous logical volume, it is difficult to extend.

Resizing File System Logical Volumes

If your users have outgrown the space originally allocated for the file system, you can increase the size of a file system by first enlarging the logical volume it resides in using the `vgextend` command, and then using the `extendfs` command to enlarge the file system contained in the logical volume.

Decreasing the size of a file system may be problematic. Based on the type of file system, you may not be able to decrease its size. However, you can create a *new* smaller file system to take its place.

For more information on resizing file system logical volumes, see "Administering File System Logical Volumes" (page 109).

Guidelines for Setting Up File System Logical Volumes

- Best system performance is achieved if you organize your volume groups by identical disk type. This means that if you create a file system that spans LVM disks, be sure that the logical volume in which the file system resides spans identical disk types.
- By default, LVM will create logical volumes on available disks, not necessarily with regard for best performance. It is possible to have a file system span two disks with different characteristics, in which case the file system performance could possibly be impaired.

  As a system administrator, you can exercise control over which physical volumes contain the physical extents of a logical volume by using the following two steps:

  1. Create a logical volume without specifying a size using the `lvcreate` command or HP SMH. When you do not specify a size, by default, no physical extents are allocated for the logical volume.
  2. Extend the logical volume (that is, allocate space) to the specific physical volumes you want to contain the file system using the `lvextend` command.
- The root or boot logical volume is limited to either 2 GB or 4 GB, depending on your processor.

Setting Up Logical Volumes for Swap

This section explains what you should consider when using logical volumes as swap devices. For information on managing your system swap space, including determining

how much and what type of swap space the system needs, see *HP-UX System Administrator's Guide: Configuration Management*.

When configured as swap, logical volumes are treated as **device swap** space. Device swap space occupies a logical volume or partition, which is typically reserved expressly for swapping purposes. This space can also be configured as a dump area (See "Guidelines for Setting Up Dump Logical Volumes" (page 38)).

Guidelines for Setting Up Swap Logical Volumes

- Interleave device swap areas for better performance.

  Two swap areas on different disks perform better than one swap area with the equivalent amount of space. This configuration allows *interleaved* swapping, which means the swap areas are written to concurrently, thus enhancing performance.

  When using LVM, set up secondary swap areas within logical volumes that are on different disks (physical volumes) using `lvextend`.

  If you have only one disk and must increase swap space, try to move the primary swap area to a larger region.

- Similar-sized device swap areas work best.

  Device swap areas should have similar sizes for best performance. Otherwise, when all space in the smaller device swap area is used, only the larger swap area is available, making interleaving no longer possible.

- To create a swap logical volume, first use the `lvcreate` command. You must set a contiguous allocation policy using the `-C y` option. For example:

  ```
  # lvcreate -C y -n swap_lvol /dev/vgnn
  ```

  Refer to *lvcreate*(1M) for more information.

- Primary swap, by default, is located on the same disk as the root file system. By default, the system kernel configuration file, `/stand/system`, contains the configuration information for primary swap.

  After creating a logical volume that will be used as primary swap, use the `lvlnboot` command with the `-s` option to update the swap information used by LVM at boot:

  ```
  # lvlnboot -s /dev/vgnn/swap_lvol
  ```

- If you are using logical volumes as secondary swap, allocate such secondary swap to reside on a disk other than the root disk for better performance.

Setting Up Logical Volumes for Dump

This section explains what you should consider when using logical volumes as dump devices. A dump area is disk space used to write an image of the core memory after a

system crash. The analysis of a core dump might be useful in troubleshooting and restoring the system to working order.

By default, the primary swap device also serves as a dump area when no other dump area is specifically designated. Although you are not required to retain primary swap as your dump area, doing so will conserve disk space. You can configure a different or multiple dump devices on your system. To do this, create a logical volume as a dump device. This device can also be used, if you want, for swap.

For information on adding, removing, or modifying dump devices, as well as configuring the dump algorithms, refer to *HP-UX System Administrator's Guide: Configuration Management*.

Guidelines for Setting Up Dump Logical Volumes

- Although dump areas can be configured within disk partitions, it is preferable to use logical volumes.
- A dump logical volume can exist only within the root volume group, that is, the volume group that contains the root logical volume.
- You can use any secondary swap logical volume as a dump area as well, provided the swap area is in the root volume group.
- To create a dump logical volume, first use the `lvcreate` command. You must set a contiguous allocation policy using the `-C y` option and disable bad block relocation using the `-r n` option. For example:

  ```
  # lvcreate -C y -r n -n dump_lvol /dev/vgnn
  ```

  Refer to *lvcreate*(1M) for more information.
- After creating a logical volume that will be used as a dump device, use the `lvlnboot` command with the `-d` option to update the dump information used by LVM. Suppose, for example, you have created a logical volume `/dev/vg00/lvol2` for use as a dump area. To update the boot information, enter:

  ```
  # lvlnboot -d /dev/vg00/lvol2
  ```

- To discontinue the use of a currently configured logical volume as a dump device, use the `lvrmboot` command with the `-d` option.

Increasing Data Redundancy Through Mirroring

NOTE: Mirroring requires an optional product, HP MirrorDisk/UX.

Mirroring is the capability of storing identical copies of data in logical volumes, preferably on separate disks. This redundancy has several advantages:

- If you mirror the root file system and swap, the operating system can tolerate a root disk failure, because valuable data is available on more than one LVM disk thus providing high availability.
- If you mirror the logical volumes used by a particular application, the application continues to run even in the event of a disk failure.
- If an I/O channel fails, LVM can recover the data from the duplicate source.
- Mirroring speeds read-intensive applications by enabling the hardware to read data from the most convenient LVM disk thus optimizing I/O.
- You can back up one copy of the data while another copy continues to run.

Disk mirroring has the obvious advantages of increased data protection and system availability, and the disadvantage of consuming twice as much disk space (or as many times more as there are mirror copies), so use disk mirroring for volatile, mission-critical data.

Single mirroring occurs when data are mapped from one logical extent to two sets of physical extents. **Double mirroring** maps each logical extent to three sets of physical extents. The number of logical extents remains constant, but the number of used physical extents (and therefore, occupied disk space) changes, depending on the number of mirrored copies.

Mirrored logical volumes must belong to the same volume group; you cannot mirror across volume groups.

This section contains the following information:

- "Controlling Mirror Write Behavior" (page 39)
- "Synchronizing a Mirrored Logical Volume" (page 42)
- "Creating and Modifying Mirrored Logical Volumes" (page 42)

If you would like to learn more about basic mirroring tasks, refer to *Disk and File Management Tasks on HP-UX* published by Prentice Hall PTR, 1997.

Controlling Mirror Write Behavior

Three policies govern how mirrored logical extents are written to physical extents: the **allocation policy**, the scheduling policy for disk writes, and the synchronization policy for crash recovery. These policies can be set using HP SMH, the `lvcreate` command, or the `lvchange` command.

Allocation Policy

Mirrored extents can be allocated on physical volumes by **strict** or **non-strict**, **contiguous** or **noncontiguous** policies. By default, the allocation policy of mirrored logical volumes is set to strict, noncontiguous.

Strict Allocation

Strict allocation requires logical extents to be mirrored to physical extents on different physical volumes. Non-strict allocation allows logical extents to be mirrored to physical extents that may be on the same physical volume. Use the `-s y` and `-s n` options to the `lvcreate` or `lvchange` commands to set strict or non-strict allocation.

CAUTION: Using non-strict allocation can reduce the redundancy offered by LVM mirroring because a logical extent could be mirrored to different physical extents on the same disk. The failure of this disk makes both copies of the data unavailable.

Contiguous Allocation

Contiguous allocation has three characteristics: the physical extents are allocated in ascending order, no gap exists between physical extents within a mirror copy, and all physical extents of a mirror copy reside in a single physical volume. Noncontiguous allocation allows logical extents to be mapped to non-consecutive physical extents. Use the `-C y` and `-C n` options to the `lvcreate` or `lvchange` commands to set contiguous or noncontiguous allocation.

NOTE: When the logical volumes allocated from the root volume group are mirrored, each must be set up with contiguous extents. That is, the physical extents used by logical volumes for root, primary swap, or dump devices must be consecutive for each mirrored copy.

Scheduling Policy

The LVM scheduler converts logical I/O requests into one or more physical I/O requests, and then schedules them for processing at the hardware level. Scheduling occurs for both mirrored and non-mirrored data.

Two I/O schedule policies are available: **parallel** and **sequential**.

Parallel Scheduling

The parallel scheduling policy is used by default with mirroring for maximum I/O performance. Parallel scheduling causes mirror operations to write simultaneously to all copies. LVM performs reads in an optimized fashion, by reading from the physical volume with the fewest outstanding I/O operations. To set the scheduling policy to parallel for a logical volume, use the `-d p` option to the `lvcreate` or `lvchange` commands.

Sequential Scheduling

The sequential scheduling policy causes mirror write operations to proceed sequentially; that is, LVM waits for one mirror write to complete before it begins the next mirror write. Likewise, LVM mirrors are read in a predefined order. On a practical level, sequential policy would be used only for extreme caution in maintaining consistency

of mirrors. To set the scheduling policy to sequential for a logical volume, use the `-d s` option to the `lvcreate` or `lvchange` commands.

Synchronization Policy

Maintaining consistency of mirrored data can be accomplished by enabling or disabling two features of your logical volume: the Mirror Write Cache and the Mirror Consistency Recovery.

Synchronization via Mirror Write Cache

The **Mirror Write Cache (MWC)** provides a fast resynchronization of data following a system crash or failure, but at a potential performance cost for routine system use.

The Mirror Write Cache keeps track of where I/O writes are occurring on the volume group, and periodically records this activity in an on-disk data structure. An extra disk write is required for every mirrored write not already recorded on the physical volume. This slows down run-time I/O write processing and degrades performance when you access the disk at random; when writing to an area of the disk that is already recorded, the performance is not impaired. Upon system reboot after crash, the operating system uses the Mirror Write Cache to resynchronize inconsistent data blocks quickly.

The frequency of extra disk writes is small for sequentially accessed logical volumes (such as database logs), but increases when access is more random. Therefore, logical volumes containing database data or file systems with few or infrequently written large files (greater than 256 KB) should not use the Mirror Write Cache when run-time performance is more important than crash-recovery time.

To enable or disable the Mirror Write Cache, use the `-M y` or `-M n` option to the `lvcreate` or `lvchange` commands.

Synchronization via Mirror Consistency Recovery

When the **LVM Mirror Consistency Recovery** is enabled, LVM does not impact run-time I/O performance. However, following a system crash, for any logical volumes using Mirror Consistency Recovery, the entire data space is resynchronized when the volume group is activated. Synchronization can be performed in the background without interfering with reboot or access; however, during this time, I/O performance and redundancy are degraded.

Synchronization with No Mirror Consistency Mechanism

When Mirror Consistency Recovery is disabled, the operating system's run-time behavior is identical to that of the previous approach. However, following a crash, LVM will *not* perform any resynchronization of data. This approach is useful for swap volumes and for volumes used by an application (such as a database) with its own means of maintaining or recovering consistent data, such as transaction log files. Note, however, that the database log files themselves can be configured as a mirrored logical volume to use the Mirror Write Cache.

To control the use of the Mirror Consistency Recovery, use the `-c y` or `-c n` option to the `lvcreate` or `lvchange` commands.

Synchronizing a Mirrored Logical Volume

At times, the data in your mirrored copy or copies of a logical volume can become out of sync, or "stale." For example, this condition might happen if LVM cannot access a disk as a result of disk power failure. Under such circumstances, for each mirrored copy to re-establish identical data, synchronization must occur. Usually, synchronization occurs automatically, although there are times when it must be done manually.

Automatic Synchronization

If you activate a volume group that is *not currently active*, either automatically at boot time or later with the `vgchange` command, then LVM by default automatically synchronizes the mirrored copies of all logical volumes with the Mirror Consistency Record policy enabled, replacing data in physical extents marked as stale with data from non-stale extents. Otherwise, no automatic synchronization occurs and manual synchronization is necessary, as described below.

LVM also automatically synchronizes mirrored data in the following cases:

- When you increase the number of mirror copies of a logical volume using the `–m` option of `lvmerge`, the newly added physical extents are synchronized.
- When a disk comes back online after experiencing a power failure.

Manual Synchronization

If you look at the status of a logical volume using `lvdisplay -v`, you can verify if the logical volume contains any stale data. You can then identify which disk contains the stale physical extents. Manually synchronize the data in one or more logical volumes using either the `lvsync` command or all logical volumes in one or more volume groups using the `vgsync` command. Refer to *lvdisplay*(1M), *vgsync*(1M), and *lvsync*(1M) for more information.

Creating and Modifying Mirrored Logical Volumes

You can configure mirroring by using either HP SMH or HP-UX commands. Whenever possible, use HP SMH.

Using HP SMH

HP SMH performs the following mirroring setup and configuration tasks:

- Creating or removing a mirrored logical volume.
- Extending or reducing the size of a mirrored logical volume.

- Splitting one copy of a mirrored logical volume into its own logical volume, and merging it back with the original.
- Configuring or changing the characteristics of logical volume mirrors. You can specify:
  - The number of mirror copies.
  - Read-only or read/write access permissions.
  - Strict (including choice of using separate physical volume groups) or non-strict allocation.
  - Contiguous allocation or noncontiguous allocation.
  - Parallel or sequential scheduling policy.
  - Mirror Write Cache, Mirror Consistency, or no recovery method.

Using HP-UX Commands

Table 2-1 summarizes the commands you need for mirror setup and configuration tasks when you do not use HP SMH. Consult section 1M of the *HP-UX Reference* for the appropriate command line options to use.

Table 2-1 HP-UX Commands Needed to Create and Configure Mirroring

Task	Commands and Options Needed
Creating a mirrored logical volume	`lvcreate -m`
Subtasks:	**Add**:
Setting strict or non-strict allocation	`-s y` or `-s n`
Setting the Mirror Write Cache method	`-M y` or `-M n`
Setting the Mirror Consistency Recovery method	`-c y` or `-c n`
Setting parallel or sequential scheduling policy	`-d p` or `-d s`
Setting contiguous allocation or noncontiguous allocation	`-C y` or `-C n`
Creating a mirror copy within separate physical volume groups	`-s g`
Removing a mirrored logical volume	`lvremove`
Increasing the number of mirror copies	`lvextend -m`
Reducing the number of mirror copies	`lvreduce -m`
Changing logical volume characteristics	`lvchange`
Subtasks:	**Add:**
Same tasks and options as for `lvcreate` as described previously.	(See previous)

Table 2-1 HP-UX Commands Needed to Create and Configure Mirroring *(continued)*

Task	Commands and Options Needed
Creating physical volume groups to mirror across separate I/O channels	One of: • `vgcreate` • `vgextend`
Designating or changing whether a physical volume will serve as a spare physical volume within the volume group	One of: • `vgextend -z y` • `vgextend -z n` • `pvchange -z y` • `pvchange -z n`

Increasing Hardware Redundancy Through Disk Sparing

NOTE: Disk sparing requires the optional product, HP MirrorDisk/UX.

If a disk containing mirrored data fails, you should replace the disk as soon as possible, as described in "Replacing a Bad Disk" (page 131). Before you replace the disk, the data in your logical volume does not have an extra mirrored copy unless you have set up 2–way mirroring. Even with 2–way mirroring, your level of safety will be reduced because of the loss of one of your two mirror copies.

To prevent this possibility, you can use one or more spare disks within each of your volume groups to serve as substitute devices in the event of disk failure. With this configuration, LVM automatically "reconfigures" the volume group so that the spare physical volume will take the place of a failed device without any intervention required. That is, a copy of the data from all the logical volumes currently on the failed disk is created on the substitute physical volume. This process is referred to as **automatic sparing**, or just **sparing**. Sparing occurs while the logical volume remains available to users. You can then schedule the replacement of the failed disk at a time of minimal inconvenience to you and your users. At such time, you would then copy the data from the spare disk back to the original disk or its replacement and return the spare disk to its role as a "standby" empty disk.

Creating a Spare Disk

To configure one or more spare physical volumes into each volume group for which you want protection against disk failure. Perform these steps before a disk failure actually occurs.

NOTE: MirrorDisk/UX is not available for shared LVM environments within a high availability cluster across more than two nodes. Because MirrorDisk/UX is required for sparing, you cannot configure sparing using the following steps within such shared LVM environments. In such cases, HP recommends that you use hardware mirroring through RAID devices. Hardware mirroring often supports its own form of sparing.

1. Use the `pvcreate` command to initialize the disk as an LVM disk. However, do not use the `-B` option because spare physical volumes cannot contain boot information.

   ```
   # pvcreate /dev/rdisk/disk3
   ```

2. Ensure the volume group has been activated.

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

3. Use the `vgextend` command with `-z y` to designate one or more physical volumes as spare physical volumes within the volume group. Alternately, you can change a physical volume with no extents currently allocated within it into a spare physical volume using the `pvchange` command with the `-z y` option.

   ```
   # vgextend -z y /dev/vg01 /dev/disk/disk3
   ```

For sparing to occur:

- All logical volumes in the volume group must have been configured with strict mirroring whereby mirrored copies are maintained on separate disks because LVM copies the data onto the spare from an undamaged disk rather than from the defective disk itself.
- At least one physical volume must be available as a “standby” spare; if your last spare is already in use as a result of a prior disk failure, it cannot serve as a currently available spare.
- The available spare must be at least as large as the failed disk.

The spare physical volume disk space will not be available for extent allocation for any other purpose than in the event of serving as a substitute disk in the event of disk failure. Therefore, its physical extents will not be included in the counts shown under `total PE` or `free PE` when examining the output of the `pvdisplay` and `vgdisplay` commands.

NOTE: If it is important to maintain comparable performance in the event of disk failure, configure a spare physical volume to each bus. However, in the event that more than one disk on the same bus fails, even with this strategy, there will be some performance impact.

The `pvdisplay` and `vgdisplay` commands provide information on whether a given physical volume is an empty standby spare or currently holding data as a spare in use, along with information on any physical volume that is currently unavailable but whose data has been spared.

Reinstating a Spare Disk

After a failed disk has been repaired or a decision has been made to replace it, follow these steps to reinstate it and return the spare disk to its former standby status:

1. Physically connect the new or repaired disk.
2. Restore the LVM configuration to the reconnected disk using `vgcfgrestore`:

   ```
   # vgcfgrestore -n /dev/vg01 /dev/rdisk/disk1
   ```

3. Ensure the volume group has been activated:

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

4. Be sure that allocation of extents is now allowed on the replaced disk:

   ```
   # pvchange -x y /dev/disk/disk1
   ```

5. Use the `pvmove` command to move the data from the spare to the replaced physical volume. As a result, the data from the spare disk is now back on the original disk or its replacement, and the spare disk is returned to its role as a standby empty disk.

   ```
   # pvmove /dev/disk/disk3 /dev/disk/disk1
   ```

Increasing Performance Through Disk Striping

Disk striping distributes logically contiguous data blocks (for example, chunks of the same file) across multiple disks, which speeds I/O throughput for large files when they are read and written sequentially (but not necessarily when access is random).

The disadvantage of disk striping is that the loss of a single disk can result in damage to many files because files are purposely spread piecemeal across two or more disks.

Consider using disk striping on file systems where large files are stored, if those files are normally read and written sequentially and I/O performance is important.

Setting Up Disk Striping

When you use disk striping, you create a logical volume that spans multiple disks, allowing successive blocks of data to go to logical extents on different disks. For example, a three-way striped logical volume has data allocated on three disks, with each disk storing every third block of data. The size of each of these blocks is referred to as the **stripe size** of the logical volume.

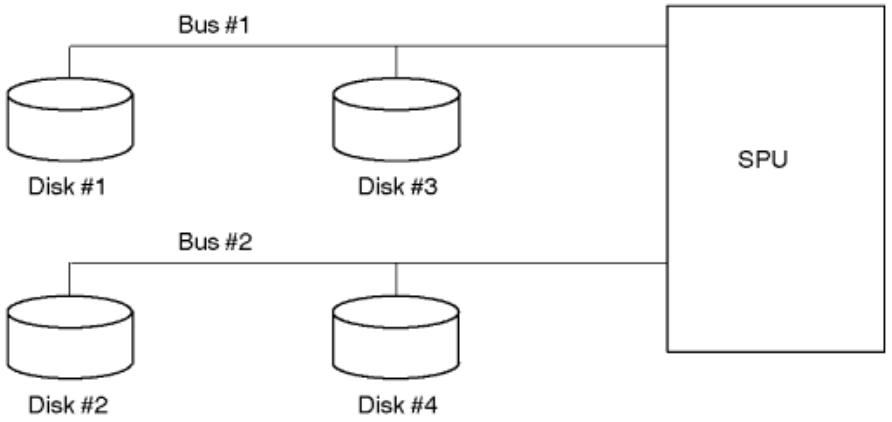
Disk striping can increase the performance of applications that read and write *large, sequentially accessed* files. Data access is performed over the multiple disks simultaneously, resulting in a decreased amount of required time as compared to the same operation on a single disk. If all of the striped disks have their own controllers, each can process data simultaneously.

You can use familiar, standard commands to manage your striped logical volumes. For example, the `lvcreate`, `diskinfo`, `newfs`, `fsck`, and `mount` commands all work with striped logical volumes.

The following guidelines, most of which apply to LVM disk usage in general, apply especially to striped logical volumes for performance reasons:

- Best performance results from a striped logical volume that spans similar disks. The more closely you match the disks in terms of speed, capacity, and interface type, the better the performance you can expect. When striping across several disks of varying speeds, performance will be no faster than that of the *slowest* disk.
- If you have more than one interface card or bus to which you can connect disks, distribute the disks as evenly as possible among them. That is, each interface card or bus should have roughly the same number of disks attached to it. You will achieve the best I/O performance when you use more than one bus and interleave the stripes of the logical volume. For example, if you have two buses with two disks on each bus, the disks should be ordered so that disk 1 is on bus 1, disk 2 is on bus 2, disk 3 is on bus 1, and disk 4 is on bus 2, as depicted in "Interleaving Disks Among Buses" (page 48).

Figure 2-2 Interleaving Disks Among Buses



- Increasing the number of disks might not necessarily improve performance because the maximum efficiency that can be achieved by combining disks in a striped logical volume is limited by the maximum throughput of the file system itself and of the buses to which the disks are attached.

To create a striped logical volume:

1. Use `pvcreate` to initialize the disks for LVM use as physical volumes.
2. Assign the physical volumes to a new or existing volume group using `vgcreate` or `vgextend`.
3. Create the striped logical volume, defining its striping characteristics using the `-i` option of `lvcreate`. The number of stripes must range from two up to the maximum number of disks in the volume group. The stripe size, the size of each of the blocks of data that make up the stripe in KB, must be a power of two in the range 4 to 32768. If you plan to use the striped logical volume for a JFS (VxFS) file system, then HP recommends using a block size of 64 KB.

   For example, suppose you want to stripe across three disks. You decide on a stripe size of 32 KB. Your logical volume size is 240 MB. To create the striped logical volume, enter the following command:

   ```
   # lvcreate -i 3 -I 32 -l 240 -n lvol1 /dev/vg01
   ```

   The `lvcreate` command automatically rounds up the size of the logical volume to a multiple of the number of disks times the extent size.

   For example, if you have three disks you want to stripe across and choose the default of 4 MB extents, even though you indicate a logical volume size of 200 (`-l 200`), `lvcreate` will create a 204 MB logical volume because 200 is not a multiple of 12.

NOTE: When you stripe across multiple disks, the striped volume size cannot exceed the capacity of the smallest disk multiplied by the number of disks used in the striping.

Determining Optimum Stripe Size

The logical volume stripe size identifies the size of each of the blocks of data that make up the stripe. You can set the stripe size to a power of two in the range 4 to 32768 (the default is 8 KB).

NOTE: The stripe size of a logical volume is not related to the physical sector size of a disk, which is typically 512 bytes.

How you intend to use the striped logical volume determines what stripe size you assign to it.

For best results:

- If you plan to use the striped logical volume for an HFS file system:

  Select the stripe size that most closely reflects the block size of the file system. The `newfs` command enables you to specify a block size when you build the file system and provides a default block size of 8 KB for HFS.

- If you plan to use the striped logical volume for a JFS (VxFS) file system:

  Use the largest available size, 64 KB. For I/O purposes, VxFS combines blocks into **extents**, which are variable in size and may be very large. The configured block size, 1 KB by default (which in any case governs only direct blocks), is not significant in this context.

- If you plan to use the striped logical volume as swap space:

  Set the stripe size to 16 KB for best performance. See "Administering Swap Logical Volumes" (page 115) for information on configuring swap.

- If you plan to use the striped logical volume as a raw data partition (for example, for a database application that uses the device directly), the stripe size should be as large or larger than the I/O size for the application.

You might need to experiment to determine the optimum stripe size for your particular situation. To change the stripe size, re-create the logical volume.

Interactions Between Mirroring and Striping

Mirroring a striped logical volume improves the read I/O performance in a same way that it does for a non-striped logical volume. Simultaneous read I/O requests targeting a single logical extent are served by two or three different physical volumes instead of

one. A striped and mirrored logical volume follows a strict allocation policy; that is, the data is always mirrored on different physical volumes.

Increasing Performance Through I/O Channel Separation

I/O channel separation is an approach to LVM configuration requiring that mirrored copies of data reside on LVM disks accessed via separate host bus adapters (interface cards) and cables. I/O channel separation achieves higher availability and better performance by reducing the number of single points of possible hardware failure. If you mirror data on two separate disks, but through one card, you are vulnerable to failure if the card fails.

You can separate I/O channels on a system with multiple HBAs, and with a single bus, by mirroring disks across different HBAs. You can further ensure channel separation by establishing a policy called **PVG-strict** allocation, which requires logical extents to be mirrored in separate physical volume groups. **Physical volume groups** are subgroups of physical volumes within a volume group.

An ASCII file, `/etc/lvmpvg`, contains all the mapping information for the physical volume group, but the mapping is not recorded on disk. Physical volume groups have no fixed naming convention; you might name them PVG0, PVG1, and so on. The `/etc/lvmpvg` file is created and updated using the `vgcreate`, `vgextend`, and `vgreduce` commands, but you can edit the file with a text editor if desired.

I/O channel separation is particularly useful for databases, because it heightens availability (LVM has more flexibility in reading data on the most accessible logical extent), resulting in better performance. If you define your physical volume groups to span I/O devices, you ensure against data loss, even if one HBA fails.

When using physical volume groups, consider using a strict PVG allocation policy for logical volumes.

Increasing Availability Through Multipathing

Your hardware might provide the capability for dual cabling (dual controllers) to the same physical volume. This capability is available if your organization has purchased an Active/Active multipathed storage enclosure or the MC/ServiceGuard product. If so, LVM can be configured with multiple paths to the same physical volume. If the primary link fails, an automatic switch to an alternate link occurs. Using multipathing increases availability.

NOTE:

As of HP-UX 11i Version 3, HP recommends using the multipathing inherent in the mass storage stack, rather than LVM's alternate links. If you want to use LVM's alternate links, you must use legacy device special files for physical volumes, and disable the mass storage stack multipathing for those legacy device special files, using the `scsimgr` command. For each multipathed disk, enter the following command:

```
# scsimgr save_attr -D /dev/rdisk/diskn -a leg_mpath_enable=0
```

Alternatively, you can disable multipathing for all legacy device files with this command:

```
# scsimgr save_attr -a leg_mpath_enable=0
```

For more information, see *scsimgr*(1M).

Setting Up Multipathing to a Physical Volume

To use an alternate link, you can create a volume group with `vgcreate`, specifying both the primary link and the alternate link device file names. Both links must represent paths to the same physical volume. (Do not run `pvcreate` on the alternate link; it must already be the same physical volume as the primary link.) When you indicate two device file names, both referring to the same disk using `vgcreate`, LVM configures the first one as the primary link and the second one as the alternate link.

For example, if a disk has two cables and you want to make one the primary link and the other an alternate link, enter:

```
# vgcreate /dev/vg01 /dev/dsk/c3t0d0 /dev/dsk/c5t0d0
```

To add an alternate link to a physical volume that is already part of a volume group, use `vgextend` to indicate the new link to the physical volume. For example, if `/dev/dsk/c2t0d0` is already part of your volume group but you want to add another connection to the physical volume, enter:

```
# vgextend /dev/vg02 /dev/dsk/c4t0d0
```

If the primary link fails, LVM automatically switches from the primary controller to the alternate controller. However, you can also tell LVM to switch to a different controller at any time by entering, for example:

```
# pvchange -s /dev/dsk/c2t1d0
```

After the primary link has recovered, LVM automatically switches back from the alternate controller to the original controller unless you previously instructed it not to by using `pvchange` as illustrated:

```
# pvchange -S n /dev/dsk/c2t2d0
```

The current links to a physical volume can be viewed using `vgdisplay` with the `-v` option.

Configuring for Optimal Recovery

Flexibility in configuration, one of the major benefits of LVM, can also be the source of problems in recovery. Here are a few guidelines to help create a configuration that minimizes recovery time.

- Keep the number of disks in the root volume group to a minimum; HP recommends using three disks, even if the root volume group is mirrored.

  Root volume groups with many disks make re-installation difficult because of the complexity of recovering LVM configurations of accessory disks within the root volume group.

  A small root volume group is quickly recovered. In some cases, you can reinstall a minimal system, restore a backup, and be back online within three hours of diagnosis and replacement of hardware (if necessary). The simplicity of recovery can trivialize a decision that otherwise might take hours to make. Another benefit is that exact match to the previous root disk layout is not a requirement.

  Three disks in the root volume group are better than two, because of quorum restrictions. With a two-disk root volume group, the loss of one disk can require you to override quorum to activate the volume group; if you must reboot to replace the disk, overriding quorum requires you to interrupt the boot process. If you have three disks in the volume group, and they are isolated from each other such that a hardware failure only affects one of them, then failure of only one disk enables the system to maintain quorum.

  There are two reasons to expand the root volume group beyond a minimal size.
  - — A very small root disk: In this case, a migration or install to a larger disk would be preferable.
  - — Providing for dump-to-swap for large memory systems: swap volumes targeted for dump must be in the root volume group. A better solution is to configure an extra dedicated disk for dump up front.

- Do not use network-based backup programs such as Omniback or Networker for basic root volume group backups. The complexity associated with these utilities can significantly delay the resumption of processing.

  HP recommends using Ignite-UX and the `make_net_recovery` command to back up and recover your root volume group.

- Be sure to have adequate documentation.

  Output from `ioscan -kf`, `vgcfgrestore -lv` for all groups and/or `vgscan -pv`, and `lvlnboot -v` are very minimal requirements. Recovery from almost any problem is possible if these, plus output from `vgdisplay -v` for all groups and `lvdisplay -v` for all volumes, are available. Extent mappings are critical to recovery of LVM volumes with corrupted headers. Additionally, output from `pvdisplay -v` for all physical volumes, although not as important, provides complete volume group information. Hard copy is not required or even necessarily practical, but accessibility during recovery is important and should be planned for.

- Many small groups are probably better than fewer large groups when configuring auxiliary volume groups.

  This issue is not as important an issue as configuration of the root volume group in recovering a down system and is probably the most debatable among all the opinions. However, reload of many dozens of gigabytes of data because one disk is missing out of a group after a root disk rebuild can be difficult. Also the scope of a catastrophic single disk failure within a group is minimized.

  Size implies complexity. The larger the number of disks in a single group, the more chances an administrator has to create an error that will affect the entire group. It is slightly simpler to identify and import smaller groups if necessary. It is certainly simpler to conceptualize and map smaller groups when required.

Preparing for the Recovery of LVM System

Following are procedures to ensure that the system data and configuration are recoverable in the event of a system failure.

1. Load any patches for LVM.
2. Use Ignite-UX to create a recovery image of your root volume group. While Ignite-UX is not intended to be used to back up all system data, you can use it with other data recovery applications to create a means of total system recovery.
3. Perform regular backups of the other important data on your system.

   Without a valid backup, you risk losing some or all of your data.

4. Regularly print out the configuration of your system.

   The configuration details stored on the system might not be accessible during a recovery. A printed copy is an invaluable reference. HP recommends printing the configuration details once a week and every time a change is made. Some of the following commands create large amounts of output. An alternative to printing them is to output the information to a file and then storing the file on tape which enables quick recovery of the information when needed. You could include this configuration file with the backup in the previous step.

The easiest way to save the configuration is to set up a `cron` job to run regularly, so that even if you do not remember to do it, the system will.

Use the following commands to get a desirable output:

`/usr/sbin/ioscan -fk`

`/usr/sbin/vgdisplay -v`

`/usr/sbin/lvlnboot -v`

`/usr/sbin/lvdisplay -v /dev/vgXX/lvYY` (for every logical volume)

`/usr/sbin/pvdisplay -v /dev/dsk/c#t#d0` (for every LVM disk)

`lp /etc/fstab`

As an alternative, you can write an intelligent script that detects any changes in the configuration and only print out those changes. An example script is included at the end of this section.

5. Back up the LVM configuration after every configuration change.

The `vgcfgbackup` command copies the LVM headers from the system area of the disk to a disk file, which, by default, resides in the `/etc/lvmconf` directory. After this information is in a disk file, it can be stored to tape during backups of the file system.

This information in this file enables you to replace the LVM headers on the disk in the event of the disk being replaced or your LVM configuration becomes corrupted.

It is important that these configuration backups are taken whenever you make a change to any part of the LVM configuration. By default all the commands perform a backup so a manual `vgcfgbackup` after each command should not be required.

This is another task that should be done on a regular basis, whether you have made changes or not. It can be done with a cron job, just before the time of a normal backup. Use this command:

```
# /usr/sbin/vgcfgbackup /dev/vgXX   (for every volume group)
```

6. Update the boot structures after every change to the root volume group.

This task is only required if you are using LVM on your boot disk. Whenever you make changes to the root volume group, which is usually named `/dev/vg00`, the BDRA on the boot disk must be updated. This is normally performed automatically by the LVM commands. To update the BDRA manually, enter the following command:

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

Example Script for LVM Configuration Recording

```
#!/usr/bin/ksh
WORKDIR=/lvmbackup # directory is regularly backed up, of course
LOG=$WORKDIR/log
SYSADM=root
if [ -f "$LOG" ]
then
    rm -f "$LOG"
fi
if [ ! -d "$WORKDIR" ]
then
    echo "missing directory $WORKDIR" exit 1
fi
cd $WORKDIR
/usr/sbin/vgdisplay -v -F > vgdisplay.new
LVMVGS=`grep vg_name vgdisplay.new | cut -d: -f1 | cut -d= -f2`
LVMPVOLS=`grep pv_name vgdisplay.new | cut -d: -f1 | cut -d= -f2 | cut -d,
-f1`
LVMLVOLS=`grep lv_name vgdisplay.new | cut -d: -f1 | cut -d= -f2`
/usr/sbin/pvdisplay -v $LVMPVOLS > pvdisplay.new
/usr/sbin/lvdisplay -v $LVMLVOLS > lvdisplay.new
/usr/sbin/lvlnboot -v > lvlnboot.new 2> /dev/null
/usr/sbin/ioscan -fk > ioscan.new
cp /etc/fstab fstab.new
for CURRENT in *new.
do
    ORIG=${CURRENT%.new}
    if diff $CURRENT $ORIG > /dev/null then
        # files are the same....do nothing
        rm $CURRENT
    else
        # files differ...make the new file the current file, move old
        # one to file.old.
        echo `date` "The config for $ORIG has changed." >> $LOG
        echo "Copy of the new $ORIG config has been printed" >> $LOG
        lp $CURRENT
        mv $ORIG ${ORIG}old.
        mv $CURRENT $ORIG
    fi
done
if [ -s "$LOG" ]
then
    mailx -s "LVM configs have changed" $SYSADM < $LOG
fi
exit 0
```

Configuring for Performance

This section provides a description of strategies to obtain the best possible performance using LVM.

The first section deals with general system resource issues that LVM impacts which might affect overall system performance. The second section covers how various

configuration decisions affect I/O performance through LVM. You can decide what combination of options will yield the best performance for your system.

General Performance Factors

These are things that affect overall system performance but not necessarily the performance of LVM.

Memory Usage

The amount of memory used by LVM is based on the values used at volume group creation time and on the number of open logical volumes. The largest portion of LVM memory is used for extent maps. The memory used is proportional to the maximum number of physical volumes multiplied by the maximum number of physical extents per physical volume for each volume group.

The other factors to be concerned with regarding these parameters are expected system growth and number of logical volumes required. You might set the volume group maximum parameters to exactly what is required on the system today. However, if you want to extend the volume group by another disk (or perhaps replace one disk with a larger disk), you will need to use the `vgmodify` command.

CPU Usage

Compared to the non-LVM case, no significant impact to system CPU usage (by observing idle time) has been observed.

As far as LVM itself is concerned, extra CPU cycles are required to perform mirror write consistency cache operations, which is the only configurable option that impacts CPU usage.

Disk Space Usage

LVM reserves some disk space on each physical volume for its own metadata. The amount of space used is proportional to the maximum values used at volume group creation time.

Internal Performance Factors

The following factors directly affect the performance of I/O through LVM.

Scheduling Policy

This factor is significant only in the mirroring case. When mirroring, the sequential scheduling policy requires more time to perform writes proportional to the number of mirrors. For instance, a logical volume with three copies of data requires three times as long to perform a write using the sequential scheduling policy compared to the parallel policy. Read requests are always directed to only one device. Under the parallel scheduling policy, LVM directs each read request to the least busy device. Under the

sequential scheduling policy, LVM directs all read requests to the device shown on the left hand side of an `lvdisplay -v` output.

Mirror Write Consistency Cache

The purpose of the Mirror Write Consistency Cache (MWC) is to provide a list of mirrored areas that might be out of sync. When a volume group is activated, LVM copies all areas with an entry in the MWC from one of the good copies to all the other copies. This process ensures that the mirrors are consistent but does not guarantee the quality of the data.

On each write request to a mirrored logical volume that uses MWC, LVM potentially introduces one extra serial disk write to maintain the MWC. Whether this condition occurs depends on the degree to which accesses are random.

The more random the accesses, the higher probability of missing the MWC. Getting an MWC entry might involve waiting for one to be available. If all the entries are currently being used by I/O in progress, a given request might have to wait in a queue of requests until one becomes available.

Two other modes of mirror consistency recovery are available: Mirror Consistency Recovery (MCR) and none. Whether you use the MWC depends on which aspect of system performance is more important to your environment: run-time or recovery-time.

Example: A customer using mirroring on a database system might choose "none" for the database logical volume because the database logging mechanism already provides consistency recovery. The logical volume used for the log would use the MWC if quick recovery time was an issue or MCR if higher run-time performance were an issue. A database log is typically used by one process and is sequentially accessed, which means it will suffer little performance degradation using MWC because the cache will be hit most of the time.

Disk Spanning

For disk areas that see the most intensive use by multiple processes, it is advantageous to spread the data space for this disk area across as many physical volumes as possible.

Disk Striping

Many have assumed that disk striping is the answer to all disk I/O performance problems, which is not the case. Applications that exhibit small, concurrent, random I/O (such as a database) see no performance gain through disk striping. Consider four disks with a stripe size of 512 bytes. Each 2 KB request is sent to all disks. One 2 KB request completes in about the same time when the disk has been striped. However, several 2 KB requests are all serialized because all the disks must seek for each request. In the non-striped case, performance might actually be better because each disk might service separate requests in parallel. (Disk striping is highly beneficial for applications with few users and large, sequential transfers.)

Number of Volume Groups

This factor is directly related to the MWC issues. Because there is only one MWC per volume group, disk space that is used for many small random write requests should be kept in distinct volume groups if possible when the MWC is being used. Aside from this, there is no other performance consideration that affects the decision regarding the number of volume groups to configure.

Physical Volume Groups

This feature can be used to enforce the separation of different mirror copies across I/O channels. It is up to the user to define the physical volume groups. This feature increases the availability by decreasing the single points of failure and provides faster I/O throughput because of less contention at the hardware level.

For example, in a system with several disk devices on each card and several cards on each bus converter, create physical volume groups so that all disks off of one bus converter are in one group and all the disks on the other are in another group. This configuration ensures that all mirrors are created with devices accessed through completely different I/O paths.

3 Administering LVM

This section contains information on the day-to-day operation of LVM. It includes the following topics:

Administration Tools

HP-UX provides two ways to manage your LVM configuration:

- **System Management Homepage (HP SMH)**: This is an HP-UX tool that provides an easy-to-use graphic user interface for performing most LVM tasks. HP SMH minimizes or eliminates the need for detailed knowledge of many administration commands, thus saving valuable time. Use HP SMH to manage your LVM configuration whenever possible, especially when first mastering a task. In particular, HP SMH can help you with the following tasks:
  - Creating or removing volume groups
  - Adding or removing disks within volume groups
  - Activating and deactivating volume groups
  - Exporting and importing volume groups
  - Creating, removing, or modifying logical volumes
  - Increasing the size of logical volumes
  - Creating or increasing the size of a file system in a logical volume
  - Creating and modifying mirrored logical volumes
  - Creating striped logical volumes
  - Splitting a mirrored logical volume and merging a mirror copy
  - Adding and removing mirror copies within mirrored logical volumes
  - Creating and modifying physical volume groups
- **LVM command-line interface**: LVM has a number of low-level user commands to perform LVM tasks, described in "Commands Needed for Physical Volume Management Tasks" (page 29), "Commands Needed for Volume Group Management Tasks" (page 29), and Commands Needed for Logical Volume Management Tasks. These are inherently more powerful—and thus dangerous—and may offer options that are not available using HP SMH. For

example, the following tasks cannot currently be done by HP SMH. For these, use the appropriate HP-UX commands:

— “Creating a Spare Disk” (page 44)
— “Reinstating a Spare Disk” (page 46)
— “Synchronizing a Mirrored Logical Volume” (page 42)
— “Moving Data to a Different Physical Volume” (page 88)
— “Replacing a Bad Disk” (page 131)

The remainder of this chapter explains how to do LVM tasks using HP-UX commands. However, HP SMH is the tool of choice for most administration work.

Displaying LVM Information

To display information about volume groups, logical volumes, or physical volumes, three commands are available. Each command supports the `-v` option to display detailed output and the `-F` option to help with scripting.

Information on Volume Groups

Use the `vgdisplay` command to show information about volume groups.

```
# vgdisplay -v vg01
-- Volume groups --
VG Name                     /dev/vg01
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      1
Open LV                     1
Max PV                      16
Cur PV                      1
Act PV                      1
Max PE per PV               1016
VGDA                        2
PE Size (Mbytes)            4
Total PE                    508
Alloc PE                    508
Free PE                     0
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0

-- Logical volumes --
LV Name                     /dev/vg01/lvol1
LV Status                   available/syncd
LV Size (Mbytes)            2032
Allocated PE                508
Used PV                     1

-- Physical volumes --
```

```
PV Name                     /dev/disk/disk42
PV Status                   available
Total PE                    508
Free PE                     0
Autoswitch                  On
```

The `vgdisplay` command is useful to verify whether the LVM configuration in memory has no problems. There should be no error messages. The status should be `available` (or `available/exclusive` for Serviceguard volume groups. All the physical volumes should be active; that is, the current number of physical volumes (`Cur PV`) should be equal to number of active physical volumes (`Act PV`). All the logical volumes should be open; that is, `Cur LV` should be equal to `Open LV`.

Information on Physical Volumes

Use the `pvdisplay` command to show information about physical volumes.

```
# pvdisplay -v /dev/disk/disk47
-- Physical volumes --
PV Name                     /dev/disk/disk47
VG Name                     /dev/vg00
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      9
PE Size (Mbytes)            4
Total PE                    1023
Free PE                     494
Allocated PE                529
Stale PE                    0
IO Timeout (Seconds)        default

-- Distribution of physical volume --
LV Name            LE of LV  PE for LV
/dev/vg00/lvol1    25        25
/dev/vg00/lvol2    25        25
/dev/vg00/lvol3    50        50

--- Physical extents ---
PE   Status   LV                  LE
0000 current  /dev/vg00/lvol1     0000
0001 current  /dev/vg00/lvol1     0001
0002 current  /dev/vg00/lvol1     0002
1021 free                         0000
1022 free                         0000
```

Stale PE should be 0.

Information on Logical Volumes

Use the `lvdisplay` command to show information about logical volumes.

```
# lvdisplay -v /dev/vg00/lvol1
-- Logical volumes --
LV Name                     /dev/vg00/lvol1
VG Name                     /dev/vg00
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               0
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            100
Current LE                  25
Allocated PE                25

Stripes 0
Stripe Size (Kbytes)        0
Bad block                   off
Allocation                  strict/contiguous

-- Distribution of logical volume --
PV Name            LE on PV  PE on PV
/dev/disk/disk42   25        25

-- Logical extents --
LE   PV1                 PE1  Status 1
0000 /dev/disk/disk42   0000 current
0001 /dev/disk/disk42   0001 current
0002 /dev/disk/disk42   0002 current
```

Common LVM Tasks

The section contains information on the following topics:

Initializing a Disk for LVM Use

WARNING! Initializing a disk using `pvcreate` results in the loss of any existing data currently on the disk.

To initialize a disk for use as a physical volume, use the following procedure. If your disk is already connected to the system, skip the first four steps:

1. Shut down and power off the system.
2. Connect the disk to the system and power supply. For detailed information and instructions on adding a particular type of disk, refer to your device documentation.
3. Power up the disk.
4. Boot the system.
5. Determine the disk's associated device file. Use the `ioscan` command with the `-f`, `-N`, and `-n` options to show the disks attached to the system and their device file names. See *ioscan*(1M) for more information.
6. Initialize the disk as a physical volume by using the `pvcreate` command. For example, enter:

   ```
   # pvcreate /dev/rdisk/disk3
   ```

   Use the *character* device file for the disk.

   If you are initializing a disk for use as a boot device, add the `-B` to `pvcreate`. The `-B` option reserves an area on the disk for a LIF volume and boot utilities. If you are creating a boot disk on an HP Integrity server, make sure the device file specifies the HP-UX partition number (2). For example, enter:

   ```
   # pvcreate -B /dev/rdisk/disk3_p2
   ```

After a disk is initialized, it is called a **physical volume**.

Creating a Volume Group

Creating a volume group requires you first to create device files to manage the volume group, and then to run the `vgcreate` command. You can create an empty volume group, or assign physical volumes to it at creation time.

1. Create a directory for the volume group. For example:

   ```
   # mkdir /dev/vgname
   ```

   By convention, *vgname* is `vg`*nn*, where *nn* is a unique number across all volume groups. However, you can choose any name up to 255 characters, as long as the name is unique.

2. Create a device file named `group` in the previously described directory with the `mknod` command.

   ```
   # mknod /dev/vgname/group c 64 0xnn0000
   ```

   The `c` following the device file name specifies that `group` is a character device file.

   The `64` is the major number for the `group` device file; it will always be `64`.

   The `0x`*nn*`0000` is the minor number for the `group` file in hexadecimal. Each particular *nn* must be a unique number across all volume groups.

   For more information on `mknod`, see *mknod*(1M); for more information on major numbers and minor numbers, refer to "Device Number Format" (page 121).

3. Create the volume group using the `vgcreate` command, specifying each physical volume to be included. For example:

   ```
   # vgcreate /dev/vgname /dev/disk/disk3
   ```

   Use the *block* device file to include each disk in your volume group. You can assign all the physical volumes to the volume group with one command, or create the volume group with a single physical volume. No physical volume can already be part of an existing volume group.

Adding a Disk to a Volume Group

Often, as new disks are added to a system, they must be added to an existing volume group rather than creating a whole new volume group. If new disks are being added for user data, such as file systems, databases, and so on, it is best not to add them to the root volume group. Instead, leave the root volume group as only the disks containing the root file system and system file systems such as `/usr`, `/tmp`, etc.

To add a disk to a volume group, initialize the disk as a physical volume by using the `pvcreate` command, as described in "Initializing a Disk for LVM Use" (page 63).

Then add the physical volume to the volume group using the `vgextend` command and the block device file for the disk. For example, enter:

```
# vgextend /dev/vgname /dev/disk/disk3
```

Removing a Disk from a Volume Group

To remove a disk from a volume group, you must first make sure the disk has no assigned physical extents, then use the `vgreduce` command:

1. Use the `pvdisplay` command as follows:

```
# pvdisplay /dev/disk/disk3
-- Physical volumes --
PV Name                     /dev/disk/disk3
VG Name                     /dev/vg00
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      9
PE Size (Mbytes)            4
Total PE                    1023
Free PE                     494
Allocated PE                529
Stale PE                    0
IO Timeout (Seconds)        default

-- Distribution of physical volume --
LV Name            LE of LV  PE for LV
/dev/vg00/lvol1    25        25
/dev/vg00/lvol2    25        25
/dev/vg00/lvol3    50        50

--- Physical extents ---
PE   Status   LV                  LE
0000 current  /dev/vg00/lvol1     0000
0001 current  /dev/vg00/lvol1     0001
0002 current  /dev/vg00/lvol1     0002
1021 free                         0000
1022 free                         0000
```

Make sure the number of free physical extents (Free PE) matches the total number of physical extents (Total PE). If they are not the same, you must do one of two things:

- Move the physical extents onto another physical volume in the volume group. See "Moving Data to a Different Physical Volume" (page 88).
- Remove the logical volumes from the disk, as described in "Removing a Logical Volume" (page 71). The logical volumes with physical extents on the disk are shown at the end of the `pvdisplay` listing.

2. After the disk no longer holds any physical extents, use the `vgreduce` command to remove it from the volume group:

```
# vgreduce /dev/vgnn /dev/disk/disk3
```

IMPORTANT: If you are using LVM pvlinks, as described in "Increasing Availability Through Multipathing" (page 50), you must run the `vgreduce` command for each link to the disk.

Creating a Logical Volume

To create a logical volume, use the following procedure:

1. Decide how much disk space the logical volume needs.

   For example, you might want to add 200 MB of device swap space, or you might be adding a new project that you expect to grow to 10 GB.

2. Find a volume group that has as much free space as you need.

   To determine if there is sufficient disk space available for the logical volume within a volume group, use the `vgdisplay` command to calculate this information. `vgdisplay` outputs data on one or more volume groups, including the physical extent size (under `PE Size (MBytes)`) and the number of available physical extents (under `Free PE`). By multiplying these two figures, you get the number of megabytes available within the volume group. Refer to *vgdisplay*(1M) for more information.

3. Create the logical volume using `lvcreate`. For example:

```
# lvcreate /dev/vgnn -L size_in_MB
```

   This command creates the logical volume `/dev/vg`*nn*`/lvol`*n* with LVM automatically assigning the *n* in `lvol`*n*.

When LVM creates the logical volume, it creates block and character device files for that logical volume and places them in the directory `/dev/vg`*nn*.

NOTE: Adding space to a logical volume does not automatically assign that space to what uses the logical volume. For example, if you want to add space to a file system contained in a logical volume, you must run `extendfs` after extending the logical volume. See "Administering File System Logical Volumes" (page 109) and "Administering Swap Logical Volumes" (page 115) for more information.

1. Decide how much more disk space the logical volume needs.

   For example, you might want to add 200 MB of swap space, or an existing project might need an additional 1 GB.

2. Find out if any space is available, using the `vgdisplay` command.

   You will see output something like this:

```
# vgdisplay
--- Volume groups ---
VG Name                     /dev/vg00
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      8
Open LV                     8
Max PV                      16
Cur PV                      1
Act PV                      1
Max PE per PV               2000
VGDA                        2
PE Size (Mbytes)            4
Total PE                    249
Alloc PE                    170
Free PE                     79
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0
```

   The `Free PE` entry indicates the number of 4 MB extents available, in this case, 79 (316 MB).

3. Extend the logical volume:

```
# /sbin/lvextend -L new_size /dev/vgnn/lvoln
```

   For example:

```
# /sbin/lvextend -L 332 /dev/vg00/lvol7
```

   increases the size of this volume to 332 MB.

Extending a Logical Volume to a Specific Disk

For performance reasons, you may want to force a logical volume to span disks. For example, suppose you want to create a 30 GB logical volume and put 10 GB on your first disk, another 10 GB on your second disk, and 10 GB on your third disk. Assuming that the extent size is the default (4 MB), the logical volume requires a total of 7680 extents. Follow this procedure:

1. After making the disks physical volumes and creating your volume group, create a logical volume named `lvol1` of size 0.

   ```
   # lvcreate -n lvol1 /dev/vg01
   ```

2. Allocate a third of the extents to the logical volume on the first physical volume.

   ```
   # lvextend -l 2560 /dev/vg01/lvol1 /dev/disk/disk7
   ```

3. Increase the total number of physical extents allocated to the logical volume for the remaining physical volumes by 2560. In each case, the additional 2560 extents are allocated to the disk specified.

   ```
   # lvextend -l 5120 /dev/vg01/lvol1 /dev/disk/disk8
   # lvextend -l 7680 /dev/vg01/lvol1 /dev/disk/disk9
   ```

When you use the `-l` option (lowercase L) of `lvextend`, you specify space in logical extents.

As another example, suppose you have two disks in a volume group, both identical models. You currently have a 24 GB logical volume that resides on only one of the disks. You want to extend the logical volume size to 40 GB, ensuring the 16 GB increase is allocated to the other disk.

Again, extend the logical volume to a specific disk.

```
# lvextend -L 40960 /dev/vg01/lvol2 /dev/disk/disk3
```

Here, when you use the `-L` option (uppercase), you are specifying space in megabytes, not logical extents.

Refer to *lvextend*(1M) for complete information on command options.

Reducing a Logical Volume

CAUTION: Before you reduce a logical volume, you must notify the user of that logical volume that its size is being reduced.

For example, before reducing a logical volume that contains a file system, *back up the file system.* Even if the file system currently occupies less space than the new (reduced) size of the logical volume, you will almost certainly lose data when you reduce the logical volume. See "Administering File System Logical Volumes" (page 109) and "Administering Swap Logical Volumes" (page 115) for the appropriate procedures for file systems and swap devices.

1. Use the `fuser` command to find out what applications are using the logical volume as follows:

   ```
   # fuser -cu /dev/vg01/lvol5
   ```

   If the logical volume is in use, ensure the underlying applications can handle the size reduction. You will likely have to stop the applications.

2. Decide on the new size of the logical volume.

   For example, if the logical volume is mounted to a file system, the new size should be greater than the space the data in the file system currently occupies. The `bdf` command will show you the size of all mounted volumes in kilobytes. The first column shows the space allocated to the volume; the second shows how much is actually being used. The new size of the logical volume should be at least a little larger than the size shown in the second column of the `bdf` output.

3. Reduce the size of the logical volume:

   ```
   # lvreduce -L 500 /dev/vg01/lvol5
   ```

   This command reduces the logical volume`/dev/vg01/lvol5` to 500 MB.

Adding a Mirror to a Logical Volume

NOTE: Mirroring requires the optional product, HP MirrorDisk/UX.

TIP: This task is easier to perform with HP SMH. In particular, HP SMH will confirm that enough disk space is available for the mirror copy and that the available space meets any allocation policy.

1. Decide how many mirror copies you want.

   For the purposes of this example, we will assume you want one mirror; that is, you will be keeping two copies of the data online, the original and one mirror copy.

2. Make sure that there is enough free space in the volume group that contains the logical volume you want to mirror.

   The volume group needs at least as much free space as the logical volume you want to mirror currently has allocated to it—that is, you will be doubling the amount of physical space this volume requires.

   If you want to use strict mirroring (which HP recommends because it keeps the mirror copy on a separate disk) this free space must be on a disk or disks not currently used by the volume you want to mirror.

3. Use the `lvextend` command with the `-m` option to add the number of additional copies you want. For example:

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

   This adds a single mirror copy of the given logical volume.

   If you want to force the mirror copy onto a specific physical volume, add it at end of the command line. For example:

   ```
   # lvextend -m 1 /dev/vg00/lvol1 /dev/disk/disk4
   ```

Removing a Mirror from a Logical Volume

To remove a mirror copy, use the `lvreduce` command, specifying the number of mirror copies you want to leave. For example, to remove all mirrors of a logical volume:

```
# lvreduce -m 0 /dev/vg00/lvol1
```

This reduces the number of mirror copies to 0, so only the original copy is left.

To remove the mirror copy from a specific disk, use `lvreduce`, and specify the disk from which to remove the mirror copy. For example:

```
# lvreduce -m 0 /dev/vg00/lvol1 /dev/disk/disk4
```

Renaming a Logical Volume

To change the name of a logical volume, rename the device files used to access it. These reside in the `/dev/vgname` directory.

1. Make sure that the logical volume has two existing device files, a block device file and a character or raw device file. They should have the same name, except that the character device file name has a leading `r`. For example, if you want to rename a logical volume in volume group `vg00` from `lvol1` to `database`, list the contents of the `/dev/vg00` directory:

```
# cd /dev/vg00
# ls -l
total 0
crw-r-----   1 root       sys  64 0x000000 Nov 16 02:49 group
brw-r-----   1 root       sys  64 0x000001 Nov 16 02:49 lvol1
brw-r-----   1 root       sys  64 0x000002 Nov 16 02:49 lvol2
brw-r-----   1 root       sys  64 0x000003 Nov 16 02:49 lvol3
brw-r-----   1 root       sys  64 0x000004 Nov 16 02:49 lvol4
crw-r-----   1 root       sys  64 0x000001 Nov 16 02:49 rlvol1
crw-r-----   1 root       sys  64 0x000002 Nov 16 02:49 rlvol2
crw-r-----   1 root       sys  64 0x000003 Nov 16 02:49 rlvol3
crw-r-----   1 root       sys  64 0x000004 Nov 16 02:49 rlvol4
```

2. Use the `mv` command to rename both files. For example:

```
# mv /dev/vg00/lvol1 /dev/vg00/database
# mv /dev/vg00/rlvol1 /dev/vg00/rdatabase
```

3. Update all references to the old name in any other files on the system. These would include `/etc/fstab` for mounted file systems or swap devices, and existing mapfiles from a `vgexport` command.

Removing a Logical Volume

CAUTION: Removing a logical volume makes its contents unavailable, and likely to be overwritten. In particular, any file system contained in the logical volume will be destroyed.

1. Make sure that the logical volume is not in use, either as a file system or as raw disk space for an application. Use the `fuser` command as follows:

```
# fuser -cu /dev/vg01/lvol5
```

If the logical volume is in use, confirm that the underlying applications no longer need it. You will likely have to stop the applications.

2. Use the `lvremove` command to remove the logical volume:

   ```
   # lvremove /dev/vg01/lvol5
   ```

You can now use this space to extend an existing logical volume or build a new logical volume.

Exporting a Volume Group

Exporting a volume group removes all data concerning the volume group from the system, while leaving the data on the disks intact. The disks of an exported volume can be physically moved or connected to another system, and the volume group can be imported there.

Exporting a volume group removes information about the volume group and its associated physical volumes from `/etc/lvmtab`, and removes the volume group's directory with device files in the `/dev` directory.

If you are planning to move the volume group to another system, use the `-m` option to `vgexport` to create a *mapfile*. This ASCII file contains the logical volume names because they are not stored on the disks. You must create a mapfile if you do not use the default names `/dev/vg`*nn*`/lvol`*n* for the logical volumes in the volume group.

1. Make sure that none of the logical volumes in the volume group are in use. This may require stopping applications using any logical volumes in the volume group, and unmounting file systems contained in the volume group.

   Use the `fuser` command on each logical volume:

   ```
   # fuser -cu /dev/vgnn/lvoln
   ```

2. Deactivate the volume group:

   ```
   # vgchange -a n vgnn
   ```

3. Use the `vgexport` command to export the volume group:

   ```
   # vgexport -v -m /tmp/vgnn.map vgnn
   ```

   If there are several disks in the volume group, use the `-s` option with `vgexport`; this option adds the volume group identifier (VGID) to the mapfile. When the volume group is imported, you can avoid specifying all the disks by name. See "Importing a Volume Group" (page 73).

When vgexport completes, all information about the volume group has been removed from the system. The disks can now be moved to a different system, and the volume group can be imported there.

Importing a Volume Group

To import a volume group, the disks must be connected to the system. In addition, you must create device files to manage the volume group, just as when the volume group was first created.

1. Create a directory for the volume group. For example:

   ```
   # mkdir /dev/vgnn
   ```

2. Create a device file named group in the previously described directory with the mknod command.

   ```
   # mknod /dev/vgnn/group c 64 0xnn0000
   ```

   The c following the device file name specifies that group is a character device file.

   The 64 is the major number for the group device file; it will always be 64.

   The 0x*nn*0000 is the minor number for the group file in hexadecimal. It is important to choose a minor number that is unique on system; if you do not choose a unique minor number vgimport will fail.

3. If the volume group has nonstandard logical volume names, you will have created a mapfile when the volume group was exported. Use the vgimport command to import the volume group:

   ```
   # vgimport -v -N -m /tmp/vgnn.map /dev/vgnn list_of_disks
   ```

   If there are several disks in the volume group and the VGID was saved in the mapfile (that is, the vgexport command was performed with the –s and –m options), you can avoid specifying all of them in the vgimport command line by using the -s option; this causes vgimport to scan all the disks on the system. Any physical volumes with a VGID matching the one in the mapfile are included automatically into the volume group.

4. Activate the volume group:

   ```
   # vgchange -a y vgnn
   ```

NOTE: If the volume group contains any multipathed disks, HP recommends using HP-UX's native multipathing that is a superset of LVM's alternate links. See "Increasing Availability Through Multipathing" (page 50) for more information.

If you want to use LVM's alternate link features, importing the volume group has several implications:

- You must omit the `-N` option to the `vgimport` command.
- The `vgimport` sets the first link found as the primary link for all physical volumes. If the links are not in the desired order after the import, use `vgreduce` and `vgextend` on the primary link for each physical volume for which you want to change the primary.
- The tunable `maxfiles` must be more than double the number of disks free.

Modifying Volume Group Parameters

When you create a volume group, you set certain characteristics of the volume group, such as the maximum number of physical extents per physical volume, the maximum number of physical volumes, and the maximum number of logical volumes. Using the `vgmodify` command, you can adjust these parameters without removing and re-creating the volume group or having to move your data.

Use the following procedure to adjust these volume group parameters:

1. Run `vgmodify` to collect information about the volume group.

   Save the output from these three commands:

   ```
   # vgmodify -o -r vgnn
   # vgmodify -v -t vgnn
   # vgmodify -v -n -t vgnn
   ```

   The `-o` option attempts to optimize the values by making full use of the existing LVM metadata space. The `-t` option reports the optimized range of settings without renumbering physical extents; the `-n` option enables renumbering of physical extents.

2. Based on the information collected in the previous step, choose new values for the volume group parameters.
3. The new values may increase the size of the volume group reserved area (VGRA) on each physical volume. The VGRA resides in the LVM header, so increasing its size may require moving the first physical extent of any user data on physical volume. Use the `pvmove` command to move the first physical extent to another location.
4. Review the values by running `vgmodify` with the new settings and the `-r` option.
5. Deactivate the volume group.

6. Commit the new values by running `vgmodify` without the `-r` option.
7. Activate the volume group. Run the `vgdisplay` command to verify the settings have been applied.

As an example, you expect to add larger disks to the volume group `vg32`. You want to increase the maximum number of physical extents per physical volume (`max_pe`) and the maximum number of physical volumes (`max_pv`). Here are the steps involved:

1. Run `vgmodify` to collect information about the volume group.

   Save the output from these three commands:

```
# vgmodify -o -r vg32
Current Volume Group settings:
                        Max LV 255
                        Max PV 16
                        Max PE per PV 1016
                        PE Size (Mbytes) 32
                        VGRA Size (Kbytes) 176
New configuration requires "max_pes" are increased from 1016 to 6652
The current and new Volume Group parameters differ.
An update to the Volume Group IS required

New Volume Group settings:
                        Max LV 255
                        Max PV 16
                        Max PE per PV 6652
                        PE Size (Mbytes) 32
                        VGRA Size (Kbytes) 896
Review complete. Volume group not modified

# vgmodify -v -t vg32
Current Volume Group settings:
                        Max LV 255
                        Max PV 16
                        Max PE per PV 1016
                        PE Size (Mbytes) 32
                        VGRA Size (Kbytes) 176
                        VGRA space (Kbytes) without PE renumbering 896
                        VGRA space (Kbytes) PE renumbering lower 32768

Volume Group optimized settings (no PEs renumbered):
max_pv(-p) max_pe(-e) Disk size (Mb)
   2          53756     1720193
   3          35836     1146753
......
   16          6652      212865
......
   213         296         9473
   255         252         8065
# vgmodify -v -n -t vg32
Volume Group configuration for /dev/vg32 has been saved in
 /etc/lvmconf/vg32.conf

Current Volume Group settings:
                Max LV 255
                Max PV 16
```

```
                Max PE per PV 1016
                PE Size (Mbytes) 32
                VGRA Size (Kbytes) 176
VGRA space (Kbytes) on Physical Volumes with extents in use:
PV                     current     -n
/dev/rdisk/disk6        896       32768
/dev/rdisk/disk5        896       32768
Summary                 896       32768
Physical Extent zero is not free on all PVs. You will not achieve these
values until the first extent is made free (see pvmove(1M)) on all the
following disks:
/dev/rdisk/disk6
/dev/rdisk/disk5

Volume Group optimized settings (PEs renumbered lower):
max_pv(-p) max_pe(-e) Disk size (Mb)
61          65535      2097152
62          65532      2097056
...
252         16048       513568
255         15868       507808
```

2. Based on the output of `vgmodify -n -t`, choose 255 for `max_pv` and 15868 for `max_pe`.
3. Since the new values require physical extent 0 to be free, use `pvmove` to move it to another location:

```
# pvmove /dev/disk/disk5:0 /dev/disk/disk5
Transferring logical extents of logical volume "/dev/vg32/lvol2"...
Physical volume "/dev/disk/disk5" has been successfully moved.
Volume Group configuration for /dev/vg32 has been saved in
 /etc/lvmconf/vg32.conf

# pvmove /dev/disk/disk6:0 /dev/disk/disk6
Transferring logical extents of logical volume "/dev/vg32/lvol1"...
Physical volume "/dev/disk/disk6" has been successfully moved.
Volume Group configuration for /dev/vg32 has been saved in
 /etc/lvmconf/vg32.conf
```

4. Preview the changes by using the `-r` option to `vgmodify`:

```
# vgmodify -p 255 -e 15868 -r -n vg32
Current Volume Group settings:
            Max LV 255
            Max PV 16
            Max PE per PV 1016
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 176
The current and new Volume Group parameters differ.
An update to the Volume Group IS required

New Volume Group settings:
```

```
            Max LV 255
            Max PV 255
            Max PE per PV 15868
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 32640

Review complete. Volume group not modified
```

5. Deactivate the volume group:

```
# vgchange -a n vg32
Volume group "vg32" has been successfully changed.
```

6. Commit the new values:

```
# vgmodify -p 255 -e 15868 -n vg32
Current Volume Group settings:
            Max LV 255
            Max PV 16
            Max PE per PV 1016
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 176
The current and new Volume Group parameters differ.
An update to the Volume Group IS required

New Volume Group settings:
            Max LV 255
            Max PV 255
            Max PE per PV 15868
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 32640
New Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf"
Old Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf.old"
Starting the modification by writing to all Physical Volumes
Applying the configuration to all Physical Volumes from
 "/etc/lvmconf/vg32.conf"
Completed the modification process.
New Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf.old"
Volume group "vg32" has been successfully changed.
```

7. Activate the volume group and verify the changes:

```
# vgchange -a y vg32
Activated volume group
Volume group "vg32" has been successfully changed.

# vgdisplay vg32
--- Volume groups ---
```

```
VG Name /dev/vg32
VG Write Access read/write
VG Status available
Max LV 255
Cur LV 0
Open LV 0
Max PV 255
Cur PV 2
Act PV 2
Max PE per PV 15868
VGDA 4
PE Size (Mbytes) 32
Total PE 1084
Alloc PE 0
Free PE 1084
Total PVG 0
Total Spare PVs 0
Total Spare PVs in use 0
```

Quiescing and Resuming a Volume Group

If you plan to use a disk management utility to create a backup image or "snapshot" of all the disks in a volume group, you must make sure that LVM is not writing to any of the disks when the snapshot is being taken; otherwise, some disks can contain partially written or inconsistent LVM metadata. To keep the volume group disk image in a consistent state, you must either deactivate the volume group or quiesce it.

Deactivating the volume group requires you to close all the logical volumes in the volume group, which can be disruptive. For example, you must unmount any file system using a logical volume in the volume group. However, temporarily quiescing the volume group enables you to keep the volume group activated and the logical volumes open during the snapshot operation, minimizing the impact to your system.

You can quiesce both read and write operations to the volume group, or just write operations. While a volume group is quiesced, the `vgdisplay` command will report the volume group access mode as "`quiesced`". The indicated I/O operations will be queued until the volume group is resumed, and commands that would modify the volume group configuration will fail immediately.

To quiesce a volume group, use the `vgchange` command with the `-Q` option:

```
# vgchange -Q mode vgnn
```

The *`mode`* parameter can be either `rw`, which blocks both read and write operations, or `w`, which permits read operations but blocks write operations.

By default, the volume group remains quiesced until it is explicitly resumed. You can specify a maximum quiesce time in seconds using the `-t` option. If the quiesce time expires, the volume group is resumed automatically. For example, to quiesce volume

group vg08 for a maximum of ten minutes (600 seconds) white permitting read operations, enter the following command:

```
# vgchange -Q w -t 600 vg08
```

To resume a quiesced volume group, use the vgchange command with the -R option:

```
# vgchange -R vgnn
```

NOTE: Individual physical volumes or logical volumes cannot be quiesced using this feature. To temporarily quiesce a physical volume, in order to disable or replace it, see "Disabling a Path to a Physical Volume" (page 98). To quiesce a logical volume, quiesce or deactivate the volume group. To provide a stable image of a logical volume without deactivating the volume group, mirror the logical volume, then split off one of the mirrors, as described in "Backing Up a Mirrored Logical Volume" (page 83).

Quiescing a volume group is not persistent across reboots.

Renaming a Volume Group

To change the name of a volume group, export it, then import it using the new name. For more detailed information on how to export and import a volume group, see "Exporting a Volume Group" (page 72) and "Importing a Volume Group" (page 73).

In the following example, the volume group vg01 is being renamed to vgdb.

1. Deactivate the volume group:

   ```
   # vgchange -a n vg01
   ```

2. Determine the minor number of the volume group's group file:

   ```
   # ls -l /dev/vg01/group
   crw-r--r-- 1 root sys 64 0x010000 Mar 28  2004 /dev/vg01/group
   ```

   For this example, the volume group number is 1.

3. Export the volume group:

   ```
   # vgexport -m vg01.map vg01
   ```

4. Create the volume group directory and group file for the volume group's new name. Since this example has a volume group number of 1, use 0x01000000 as the group file's minor number:

   ```
   # mkdir /dev/vgdb
   # mknod /dev/vgdb/group c 64 0x010000
   ```

5. Import the volume group under its new name:

   ```
   # vgimport -m vg01.map /dev/vgdb
   ```

6. Back up the volume group configuration information:

   ```
   # vgcfgbackup /dev/vgdb
   ```

7. Activate the volume group:

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

8. Remove saved configuration information based on the old volume group name:

   ```
   # rm /etc/lvmconf/vg01.conf
   ```

9. Update all references to the old name in any other files on the system. These would include `/etc/fstab` for mounted file systems or swap devices, and existing mapfiles from a `vgexport` command.

Splitting a Volume Group

You can use the `vgchgid` to split an existing volume group into two or more volume groups, provided that the physical volumes to be split are self-contained; in other words, any logical volumes on the physical volumes must be wholly contained on those physical volumes. For example, a splittable volume group could have logical volumes 1, 2, and 3 on physical volumes 0 and 1, and logical volumes 4, 5, and 6 on physical volumes 2, 3, 4, and 5.

In this example, volume group `vgold` contains physical volumes `/dev/disk/disk0` through `/dev/disk/disk5`. Logical volumes `lvol1`, `lvol2`, and `lvol3` are on physical volumes `/dev/disk/disk0` and `/dev/disk/disk1`, and logical volumes `lvol4`, `lvol5`, and `lvol6` are on the remaining physical volumes. To keep `/dev/disk/disk0` and `/dev/disk/disk1` in `vgold` and split the remaining physical volumes into a new volume group named `vgnew`, you could use the following procedure:

1. Deactivate the volume group:

   ```
   # vgchange -a n vgold
   ```

2. Export the volume group:

   ```
   # vgexport vgold
   ```

3. Change the VGID on the physical volumes to be assigned to the new volume group:

```
# vgchgid -f /dev/rdisk/disk2 /dev/rdisk/disk3 \
     /dev/rdisk/disk4 /dev/rdisk/disk5
```

4. Create the volume group directory and `group` file for the old volume group:

```
# mkdir /dev/vgold
# mknod /dev/vgold/group c 64 0xm0000
```

5. Create the volume group directory and `group` file for the new volume group:

```
# mkdir /dev/vgnew
# mknod /dev/vgnew/group c 64 0xn0000
```

6. Import the physical volumes in the old volume group:

```
# vgimport /dev/vgold /dev/rdisk/disk0 /dev/rdisk/disk1
```

7. Import the physical volumes in the new volume group:

```
# vgimport /dev/vgnew /dev/rdisk/disk2 /dev/rdisk/disk3 \
     /dev/rdisk/disk4 /dev/rdisk/disk5
```

8. Activate the volume groups. Disable quorum checks for the old volume group, because it is missing over half its disks:

```
# vgchange -a y -q n /dev/vgold
# vgchange -a y /dev/vgnew
```

9. The logical volumes are currently defined in both volume groups. Remove the duplicate logical volumes from the volume group that no longer contains them:

```
# lvremove -f vgold/lvol4 vgold/lvol5 vgold/lvol6
# lvremove -f vgnew/lvol1 vgnew/lvol2 vgnew/lvol3
```

10. The physical volumes are currently defined in both volume groups. Remove the missing physical volumes from both volume groups:

```
# vgreduce -f vgold
# vgreduce -f vgnew
```

11. Enable quorum checks for the old volume group:

```
# vgchange -a y -q y /dev/vgold
```

On completion, the original volume group contains three logical volumes (`lvol1`, `lvol2`, and `lvol3`) with physical volumes `/dev/disk/disk0` and `/dev/disk/disk1`. The new volume group `vgnew` contains three logical volumes

(lvol4, lvol5, and lvol6) across physical volumes /dev/disk/disk2, /dev/disk/disk3, /dev/disk/disk4, and /dev/disk/disk5.

Removing a Volume Group

TIP: It is easier to export a volume group than to remove it, because removing the volume requires that you remove all the logical volumes and physical volumes from the volume group before running vgremove. In addition, exporting the volume group leaves the LVM information on the disks untouched, which is an advantage if you want to re-import the volume group later. For the procedure to export a volume group, see "Exporting a Volume Group" (page 72).

Removing a volume group is very similar to exporting it. The only difference is that you must first remove all the logical volumes and physical volumes from the volume group.

1. Back up all user data.
2. Find the names of all logical volumes and physical volumes in the volume group.

   ```
   # vgdisplay -v /dev/vgnn
   ```

3. Make sure that none of those logical volumes are in use. This may require stopping applications using any logical volumes in the volume group, and unmounting file systems contained in the volume group.

   Use the fuser command on each logical volume:

   ```
   # fuser -cu /dev/vgnn/lvoln
   ```

4. Remove each of the logical volumes, as described in "Removing a Logical Volume" (page 71):

   ```
   # lvremove /dev/vgnn/lvoln
   ```

5. Remove all but one of the physical volumes, as described in "Removing a Disk from a Volume Group" (page 65):

   ```
   # vgreduce /dev/vgnn /dev/disk/diskn
   ```

6. Use the vgremove command to remove the volume group:

   ```
   # vgremove vgnn
   ```

Backing Up a Mirrored Logical Volume

NOTE: Mirroring requires the optional product, HP MirrorDisk/UX.

You can split a mirrored logical volume into two logical volumes to perform a backup on an offline copy while the other copy stays online. When you complete the activity on the offline copy, you can merge the two logical volumes back into one. In order to bring the two copies back in synchronization, LVM updates the physical extents in the offline copy based on changes made to the copy that remained in use.

You can use HP SMH to split and merge logical volumes, or use the `lvsplit` and `lvmerge` commands.

The following procedure demonstrates how to back up a mirrored logical volume containing a file system, using `lvsplit` and `lvmerge`:

1. Split the logical volume `/dev/vg00/lvol1` into two separate logical volumes:

   ```
   # vgsplit /dev/vg00/lvol1
   ```

   This creates the new logical volume `/dev/vg00/lvol1b`. The original logical volume `/dev/vg00/lvol1` remains online.

2. Perform a file system consistency check on the logical volume to be backed up:

   ```
   # fsck /dev/vg00/lvol1b
   ```

3. Mount the file system:

   ```
   # mkdir /backup_dir
   # mount /dev/vg00/lvol1b /backup_dir
   ```

4. Perform the backup using the utility of your choice.
5. Unmount the file system:

   ```
   # umount /backup_dir
   ```

6. Merge the split logical volume back with the original logical volume:

   ```
   # lvmerge /dev/vg00/lvol1b /dev/vg00/lvol1
   ```

Backing Up and Restoring Volume Group Configuration

It is important that volume group configuration information be saved whenever you make *any* change to the configuration such as:

- Adding or removing disks to a volume group
- Changing the disks in a root volume group

- Creating or removing logical volumes
- Extending or reducing logical volumes

Unlike fixed disk partitions or non-partitioned disks that begin and end at known locations on a given disk, each volume group configuration is unique, changes at times, and likely uses space on several disks.

If you back up your volume group configuration, you will be able to restore a corrupted or lost LVM configuration in the event of a disk failure or corruption of your LVM configuration information (for example, through the accidental or incorrect use of commands such as `newfs` or `dd`).

The `vgcfgbackup` command is used to create or update a backup file containing the volume group configuration; it *does not back up the data within your logical volumes*. To simplify the backup process, `vgcfgbackup` is invoked automatically by default whenever you make a configuration change as a result of using any of the following commands:

`lvchange`	`lvcreate`	`lvextend`	`lvlnboot`	`lvmerge`
`lvreduce`	`lvremove`	`lvrmboot`	`lvsplit`	`vgcreate`
`pvchange`	`pvmove`	`vgextend`	`vgmodify`	`vgreduce`

You can display LVM configuration information previously backed up with `vgcfgbackup` or restore it using `vgcfgrestore`.

By default, `vgcfgbackup` saves the configuration of a volume group to the file `/etc/lvmconf/`*`volume_group_name`*`.conf`.

If you choose, you can run `vgcfgbackup` at the command line, saving the backup file in any directory you indicate. If you do, first run `vgdisplay` with the `-v` option to ensure that the all logical volumes in the volume group are shown as `available/syncd`. If so, then run:

```
# vgcfgbackup -f pathname/filename volume_group_name
```

If you use a non-default volume group configuration file, be sure to take note of and retain its location. Refer to *vgcfgbackup*(1M) for information on command options. Make sure backups of the root volume group are on the root file system, in case these are required during recovery.

To run `vgcfgrestore`, the physical volume must be detached. If all the data on the physical volume is mirrored and the mirror copies are current and available, you can temporarily detach the physical volume using `pvchange`, perform the `vgcfgrestore`, and re-attach the physical volume. For example, to restore volume group configuration data for `/dev/disk/disk5`, a disk in the volume group `/dev/vgsales`, enter:

```
# pvchange -a n /dev/disk/disk5
# vgcfgrestore -n /dev/vgsales /dev/rdisk/disk5
# pvchange -a y /dev/disk/disk5
```

If the physical volume is not mirrored or the mirror copies are not current and available, you must deactivate the volume group with `vgchange`, perform the `vgcfgrestore`, and activate the volume group:

```
# vgchange -a n /dev/vgsales
# vgcfgrestore -n /dev/vgsales /dev/rdisk/disk5
# vgchange -a y /dev/vgsales
```

These examples restore the LVM configuration to the disk from the default backup location in `/etc/lvmconf/vgsales.conf`.

Refer to *vgcfgrestore*(1M) for information on command options.

Moving and Reconfiguring Your Disks

This section contains information on the following topics:

There are occasions when you might need to:

- Move the disks in a volume group to different hardware locations on a system.
- Move entire volume groups of disks from one system to another.

CAUTION: Moving a disk that is part of your root volume group is not recommended. Refer to *Configuring HP-UX for Peripherals* for more information.

The file `/etc/lvmtab` contains information about the mapping of LVM disks on a system to volume groups, that is, volume group names and lists of the physical volumes included in volume groups. When you do either of the previous tasks, the LVM configuration file, `/etc/lvmtab`, must be changed to reflect the new hardware locations and device files for the disks. However, you cannot edit this file directly because it is not a text file. Instead, you must use `vgexport` and `vgimport` to reconfigure the volume groups. This action results in the configuration changes being recorded in the `/etc/lvmtab` file.

Moving Disks Within a System

There a two procedures for moving the disks in a volume group to different hardware locations on a system. Your choice of procedure depends on whether you use persistent or legacy device files for your physical volumes; the types of device files are described in "Legacy Device Files versus Persistent Device Files" (page 25).

If Your LVM Configuration Uses Persistent Device Files

1. Be sure that you have an up-to-date backup for both the data within the volume group and the volume group configuration.
2. Deactivate the volume group by entering:

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

3. Physically move your disks to their desired new locations.
4. Activate the volume group:

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

If Your LVM Configuration Uses Legacy Device Files

The names of legacy device files change when the hardware paths to their physical devices change. Therefore, you must update the LVM configuration with the new legacy device files. Do this by exporting and importing the volume group to use the new legacy device files. The update procedure follows:

1. Be sure that you have an up-to-date backup for both the data within the volume group and the volume group configuration.
2. Deactivate the volume group by entering:

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

3. Remove the volume group entry from `/etc/lvmtab` and its associated device files from the system by entering:

   ```
   # vgexport -v -s -m /tmp/vgnn.map /dev/vgnn
   ```

4. Physically move your disks to their desired new locations.
5. To view the new locations, enter:

   ```
   # vgscan -v
   ```

6. Add the volume group entry back to /etc/lvmtab and the associated device files back to the system:
   a. Create a new directory for the volume groups with mkdir.
   b. Create a group file in the directory described previously with mknod.
   c. Issue the vgimport command:

      # **vgimport -v -s -m /tmp/vg*nn*.map /dev/vg*nn***

7. Activate the newly imported volume group:

   # **vgchange -a y /dev/vg*nn***

8. Back up the volume group configuration:

   # **vgcfgbackup /dev/vg*nn***

Moving Disks Between Systems

The procedure for moving the disks in a volume group to different hardware locations on a different system is a matter of exporting the volume group from one system, physically moving the disks to the other system, and importing the volume group there. The procedures for exporting and importing a volume are described in "Exporting a Volume Group" (page 72) and "Importing a Volume Group" (page 73). They are illustrated in the following example.

NOTE: If the volume group contains any multipathed disks, see the note under "Importing a Volume Group" (page 73).

Suppose you want to move the three disks in the volume group /dev/vg_planning to another system. Follow these steps:

1. Make the volume group and its associated logical volumes unavailable to users. If any of the logical volumes contain a file system, the file system must be unmounted. If any of the logical volumes are used as secondary swap, disable swap and reboot the system; for information on secondary swap, see *HP-UX System Administrator's Guide: Configuration Management*.

   # **vgchange -a n /dev/vg_planning**

2. Use *vgexport*(1M) to remove the volume group information from the /etc/lvmtab file. You can first preview the actions of vgexport with the -p option.

   # **vgexport -p -v -s -m /tmp/vg_planning.map /dev/vg_planning**

With the -m option, you can specify the name of a map file that will hold the information that is removed from the /etc/lvmtab file. This file is important because it will contain the names of all logical volumes in the volume group.

You will use this map file when you set up the volume group on the new system.

If the preview is satisfactory, run the command without -p.

```
# vgexport -v -s -m /tmp/vg_planning.map /dev/vg_planning
```

The vgexport command removes the volume group from the system and creates the /tmp/vg_planning.map file.

3. Connect the disks to the new system and copy the file /tmp/vg_planning.map to the new system.
4. On the new system, create a new volume group directory and group file:

   ```
   # mkdir /dev/vg_planning
   ```

   When you create the group file, specify a minor number that reflects the volume group number. (Volume group numbering starts at 00; the volume group number for the fifth volume group, for example, is 04.)

   ```
   # mknod /dev/vg_planning/group c 64 0x040000
   ```

5. Run the ioscan command to get device file information about the disks:

   ```
   # ioscan -funN -C disk
   ```

6. Issue the vgimport command. To preview, use the -p option.

   ```
   # vgimport -p -N -v -s -m /tmp/vg_planning.map /dev/vg_planning
   ```

   To actually import the volume group, issue the command again, omitting the -p option.
7. Activate the newly imported volume group:

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

Moving Data to a Different Physical Volume

You can use the pvmove command to move data contained in logical volumes from one disk to another disk or to move data between disks within a volume group.

For example, you might want to move only the data from a specific logical volume from one disk to another to use the vacated space on the first disk for some other purpose. To move the data in logical volume /dev/vg01/markets from the disk /dev/disk/disk4 to the disk /dev/disk/disk7, enter:

```
# pvmove -n /dev/vg01/markets /dev/disk/disk4 /dev/disk/disk7
```

On the other hand, you might prefer to move all the data contained on one disk, regardless of which logical volume it is associated with, to another disk within the same volume group. You might want to do this, for example, so you can remove a disk from a volume group. You can use `pvmove` to move the data to other specific disks you choose or let LVM move the data to appropriate available space within the volume group, subject to any mirroring allocation policies.

To move all data off disk `/dev/dsk/disk3` and relocate it at the destination disk `/dev/disk/disk5`, enter:

```
# pvmove /dev/disk/disk3 /dev/disk/disk5
```

To move all data off disk `/dev/disk/disk3` and let LVM transfer the data to available space within the volume group, enter:

```
# pvmove /dev/disk/disk3
```

In each of the previous instances, if space does not exist on the destination disk, the `pvmove` command will not succeed.

NOTE: The `pvmove` command is not an atomic operation, and moves data extent by extent. If `pvmove` is abnormally terminated by a system crash or `kill -9`, the volume group can be left in an inconsistent configuration showing an additional pseudo mirror copy for the extents being moved. You can remove the extra mirror copy using the `lvreduce` command with the `-m` option on each of the affected logical volumes; there is no need to specify a disk.

Modifying Physical Volume Characteristics

The `vgmodify` command allows you to modify a volume group to adapt to changes in physical volumes. In particular, you can adjust the volume group to recognize size changes in physical volumes, and you can change a physical volume type between bootable and non-bootable.

Recognizing Size Changes

Disk arrays typically allow a LUN to be resized. If you increase the size of a LUN, use the following procedure to incorporate the additional space into the volume group:

1. Increase the LUN size using the instructions for the array.
2. Run `vgmodify` to detect any physical volume size changes. It will also report whether all of the space can be made available to the volume group.
3. If `vgmodify` reports that the maximum number of physical extents per physical volume (`max_pe`) is too small to accommodate the new size, use `vgmodify` with

the -t and -n options to determine a new value for max_pe, as described in "Modifying Volume Group Parameters" (page 74).

4. Review the values by running vgmodify with the new settings and the -r option.
5. Deactivate the volume group.
6. Commit any new value of max_pe and update the physical volume information by running vgmodify without the -r option.
7. Activate the volume group. Run the vgdisplay and pvdisplay commands to verify that the increased space is available.

As an example, you want to increase the size of the physical volume /dev/rdisk/disk6 from 4 GB to 100000000 KB. Here are the steps involved:

1. Increase the LUN size using the instructions for the disk array.
2. Run vgmodify with the -v and -r options to check whether any disks have changed in size and whether all of the space on the physical volumes can be utilized.

```
# vgmodify -v -r vg32
Current Volume Group settings:
        Max LV 255
        Max PV 16
        Max PE per PV 1016
        PE Size (Mbytes) 32
        VGRA Size (Kbytes) 176

/dev/rdisk/disk6 Warning: Max_PE_per_PV for the volume group
      (1016) too small for this PV (3051).
Using only 1016 PEs from this physical volume.

"/dev/rdisk/disk6" size changed from 4194304 to 100000000kb

An update to the Volume Group IS required

New Volume Group settings:
        Max LV 255
        Max PV 16
        Max PE per PV 1016
        PE Size (Mbytes) 32
        VGRA Size (Kbytes) 176

Review complete. Volume group not modified
```

The expanded physical volume requires 3051 physical extents to use all its space, but the current max_pe value limits this to 1016.

3. Run vgmodify -t, with and without the -n, to determine the optimal values for max_pv and max_pe.

```
# vgmodify -t vg32
       Current Volume Group settings:
        Max LV 255
```

```
Max PV 16
Max PE per PV 1016
PE Size (Mbytes) 32
VGRA Size (Kbytes) 176
VGRA space (Kbytes) without PE renumbering 896
VGRA space (Kbytes) PE renumbering lower 32768
Volume Group optimized settings (no PEs renumbered):

max_pv(-p) max_pe(-e) Disk size (Mb)
    2         53756     1720193
    3         35836     1146753
    4         26876     860033
    5         21500     688001
    6         17916     573313
    7         15356     491393
    8         13308     425857
    9         11772     376705
    10        10748     343937
    11         9724     311169
    12         8956     286593
    13         8188     262017
    14         7676     245633
    15         7164     229249
    16         6652     212865
    17         6140     196481
    18         5884     188289
    19         5628     180097
    20         5372     171905
    21         5116     163713
    22         4860     155521
    23         4604     147329
    24         4348     139137
    26         4092     130945
    28         3836     122753
    30         3580     114561
    32         3324     106369
    35         3068     98177
    38         2812     89985
    42         2556     81793
    46         2300     73601
    52         2044     65409
    60         1788     57217
    70         1532     49025
    84         1276     40833
    105        1020     32641
    140         764     24449
    141         632     20225
    210         508     16257
    211         456     14593
    212         376     12033
```

```
            213         296     9473
            255         252     8065
```

The table shows that without renumbering physical extents, a `max_pv` of 35 or lower would permit a `max_pe` sufficient to accommodate the increased physical volume size.

```
# vgmodify -v -t -n vg32
Volume Group configuration for /dev/vg32 has been saved in
 /etc/lvmconf/vg32.conf
    Current Volume Group settings:
    Max LV 255
    Max PV 16
    Max PE per PV 1016
    PE Size (Mbytes) 32
    VGRA Size (Kbytes) 176
    VGRA space (Kbytes) on all Physical Volumes:
                PV                current         -n
        /dev/rdisk/disk6            896             32768
        /dev/rdisk/disk5            896             32768
        Summary                     896             32768

Volume Group optimized settings (PEs renumbered lower):
    max_pv(-p)  max_pe(-e) Disk size (Mb)
        61        65535     2097152
        62        65532     2097056
        63        64252     2056096
    ........
        251       16124     516000
        252       16048     513568
        255       15868     507808
```

The table shows that if physical extents are renumbered, all values of `max_pv` permit a `max_pe` large enough to accommodate the increased physical volume size.

For this example, select a `max_pv` of 10, which permits a `max_pe` value of 10748.

4. Preview the changes by using the `-r` option to `vgmodify`:

```
# vgmodify -p 10 -e 10748 -r vg32
Current Volume Group settings:
                Max LV 255
                Max PV 16
                Max PE per PV 1016
                PE Size (Mbytes) 32
                VGRA Size (Kbytes) 176
The current and new Volume Group parameters differ.
"/dev/rdisk/disk6" size changed from 4194304 to 100000000kb
An update to the Volume Group IS required
```

```
New Volume Group settings:
                Max LV 255
                Max PV 10
                Max PE per PV 10748
                PE Size (Mbytes) 32
                VGRA Size (Kbytes) 896
Review complete. Volume group not modified
```

5. Deactivate the volume group:

```
# vgchange -a n vg32
Volume group "vg32" has been successfully changed.
```

6. Commit the new values:

```
# vgmodify -p 10 -e 10748 vg32
Current Volume Group settings:
                Max LV 255
                Max PV 16
                Max PE per PV 1016
                PE Size (Mbytes) 32
                VGRA Size (Kbytes) 176
The current and new Volume Group parameters differ.
"/dev/rdisk/disk6" size changed from 4194304 to 100000000kb
An update to the Volume Group IS required

New Volume Group settings:
                Max LV 255
                Max PV 10
                Max PE per PV 10748
                PE Size (Mbytes) 32
                VGRA Size (Kbytes) 896
New Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf"
Old Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf.old"
Starting the modification by writing to all Physical Volumes
Applying the configuration to all Physical Volumes from
 "/etc/lvmconf/vg32.conf"
Completed the modification process.
New Volume Group configuration for "vg32" has been saved in
 "/etc/lvmconf/vg32.conf.old"
Volume group "vg32" has been successfully changed.
```

7. Activate the volume group and verify the changes:

```
# vgchange -a y vg32
Activated volume group
Volume group "vg32" has been successfully changed.

# vgdisplay vg32
```

```
--- Volume groups ---
VG Name /dev/vg32
VG Write Access read/write
VG Status available
Max LV 255
Cur LV 0
Open LV 0
Max PV 10
Cur PV 2
Act PV 2
Max PE per PV 10748
VGDA 4
PE Size (Mbytes) 32
Total PE 3119
Alloc PE 0
Free PE 3119
Total PVG 0
Total Spare PVs 0
Total Spare PVs in use 0
```

CAUTION: This procedure can also be used when the size of a physical volume is decreased. However, there are limitations:

- To avoid data corruption, the size of the LUN (in the disk array) must be reduced only *after* `vgmodify` has successfully changed the volume group.
- The volume group must be deactivated before attempting any reduction. If you reduce the size of the LUN while the volume group is activated, LVM marks the physical volume as unavailable.
- If the physical volume has any allocated physical extents beyond the target size, `vgmodify` will print an error message and exit without changing the physical volume. In this case, you must be prepared to restore the LUN to its original size (ensuring the same disk space is allocated).

Changing Physical Volume Types

When a physical volume is initialized for LVM use, it can be made bootable or non-bootable. Bootable physical volumes require additional space in their LVM metadata for boot utilities and information. If a physical volume was inadvertently initialized as bootable, you can convert the disk to a non-bootable disk and reclaim LVM metadata space.

CAUTION: The boot volume group requires at least one bootable physical volume. Do not convert all of the physical volumes in the boot volume group to non-bootable, or your system will not boot.

Use the following procedure to change a disk type from bootable to non-bootable:

1. Use vgcfgrestore to determine if the volume group contains any bootable disks.
2. Run vgmodify twice: once with the -B n and once without it. Compare the available values for max_pe and max_pv.
3. Choose new values for max_pe and max_pv. Review the values by running vgmodify with the new settings and the -r option.
4. Deactivate the volume group.
5. Commit the changes by running vgmodify without the -r option.
6. Activate the volume group. Run the vgcfgrestore or pvdisplay commands to verify that the disk type has changed.

For example, if you want to convert any bootable disks in volume group vg, use the following procedure:

1. Check if any physical volumes in vg01 are bootable:

```
# vgcfgrestore -l -v -n vg01
Volume Group Configuration information in "/etc/lvmconf/vg01.conf"

VG Name /dev/vg01
---- Physical volumes : 1 ----
PV Type Size (kb) Start (kb) PVkey
c2t1d0 Bootable 35566480 2912 0

max_pv 16 max_pe 1085 max_lv 255
```

2. Run vgmodify with the -t option to determine which values of max_pe and max_pv are available. Compare the values if the disk is made non-bootable, and if it is not:

```
# vgmodify -t -B n vg01 /dev/rdsk/c2t1d0
Current Volume Group settings:
            Max LV 255
            Max PV 16
            Max PE per PV 1085
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 208
VGRA space (Kbytes) without PE renumbering 2784
VGRA space (Kbytes) PE renumbering lower 32768
Volume Group optimized settings (no PEs renumbered):

max_pv(-p) max_pe(-e) Disk size (Mb)

5          65535         2097122
6          56828         1818498
.......
255         1276           40834


# vgmodify -t vg01
Current Volume Group settings:
```

```
            Max LV 255
            Max PV 16
            Max PE per PV 1085
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 208
VGRA space (Kbytes) without PE renumbering 768
VGRA space (Kbytes) PE renumbering lower 768
Volume Group optimized settings (no PEs renumbered):

max_pv(-p) max_pe(-e) Disk size (Mb)

1          65535       2097120
2          45820       1466240
.....
255          252          8064
```

If you change the disk type, the VGRA space available increases from 768 KB to 2784KB (if physical extents are not renumbered) or 32768 KB (if physical extents are renumbered). Changing the disk type also permits a larger range of `max_pv` and `max_pe`. For example, if `max_pv` is 255, the bootable disk can only accommodate a disk size of 8064 MB, but after conversion to non-bootable, it can accommodate a disk size of 40834 MB.

3. For this example, select a `max_pv` value of 6, which permits a `max_pe` value of 56828. Preview the changes by using the `-r` option to `vgmodify`:

```
# vgmodify -r -p 6 -e 56828 -B n vg01 /dev/rdsk/c2t1d0
Current Volume Group settings:
            Max LV 255
            Max PV 16
            Max PE per PV 1085
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 208
The current and new Volume Group parameters differ.
An update to the Volume Group IS required

New Volume Group settings:
            Max LV 255
            Max PV 6
            Max PE per PV 56828
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 2784
Review complete. Volume group not modified
```

4. Deactivate the volume group:

```
# vgchange -a n vg01
Volume group "vg01" has been successfully changed.
```

5. Commit the new values:

```
# vgmodify -p 6 -e 56828 -B n vg01 /dev/rdsk/c2t1d0
Current Volume Group settings:
            Max LV 255
            Max PV 16
            Max PE per PV 1085
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 208
The current and new Volume Group parameters differ.
An update to the Volume Group IS required

New Volume Group settings:
            Max LV 255
            Max PV 6
            Max PE per PV 56828
            PE Size (Mbytes) 32
            VGRA Size (Kbytes) 2784
New Volume Group configuration for "vg01" has been saved in
 "/etc/lvmconf/vg01.conf"
Old Volume Group configuration for "vg01" has been saved in
 "/etc/lvmconf/vg01.conf.old"

Starting the modification by writing to all Physical Volumes
Applying the configuration to all Physical Volumes from
 "/etc/lvmconf/vg01.conf"
Completed the modification process.
New Volume Group configuration for "vg01" has been saved in
 "/etc/lvmconf/vg01.conf.old"
Volume group "vg01" has been successfully changed.
```

6. Activate the volume group and verify the changes:

```
# vgchange -a y vg01
Activated volume group
Volume group "vg01" has been successfully changed.

# vgcfgbackup vg01
Volume Group configuration for /dev/vg01 has been saved in
 /etc/lvmconf/vg01.conf

# vgcfgrestore -l -v -n vg01
Volume Group Configuration information in "/etc/lvmconf/vg01.conf"
VG Name /dev/vg01

---- Physical volumes : 1 ----
PV     Type     Size (kb) Start (kb) PVkey
c2t1d0 Non-Boot 35566480   2912       0

max_pv 6 max_pe 56828 max_lv 255
```

Disabling a Path to a Physical Volume

WARNING! This procedure only disables *LVM's* use of the link. The `pvchange` command will not prevent diagnostics or an application from accessing the physical volume.

By default, the mass storage stack uses all the paths available to access a physical volume, independently of the paths configured in LVM. Disabling a path in LVM does not prevent the native multipathing from using that path. Use the `scsimgr` command to disable I/O along a path or to disable the native multipathing.

You can temporarily disable LVM's use of one or all of the physical paths to a physical volume using the `pvchange` command. Disabling a path, also known as **detaching the link**, causes LVM to close that path to the device and stop using it; this can be useful if you want to guarantee that a link is idle, such as when you are running diagnostics on an I/O card, replacing an I/O card, or replacing the disk containing the physical volume.

Detaching a link to a physical volume is intended to be a temporary operation, not a permanent one. If you want to permanently remove a link or physical volume from the volume group, use `vgreduce` instead.

To detach a link to a physical volume, use the `-a` option to `pvchange`. For example, to disable the link through the device `/dev/disk/disk33`, enter:

```
# pvchange -a n /dev/disk/disk33
```

If you are using LVM's alternate links for multipathed disks, each link uses a different legacy device files. In that situation, to detach all links to a physical volume, use `N` as the argument to the `-a` option:

```
# pvchange -a N /dev/dsk/c5t0d0
```

Detaching one or more links to a physical volume will not necessarily cause LVM to stop using that physical volume entirely. If the detached link is the primary path to the device, LVM will begin using any available alternate link to it. LVM will only stop using the physical volume when all the links to it are detached.

If all the links to a device are detached, the associated physical volume will be unavailable to the volume group. The links remain associated with the volume group but LVM will not send any I/O requests to the physical volume until it is reattached. This means that the data on that physical volume will be temporarily unavailable; consequently, you as the administrator must make sure that any availability requirements for that data can be satisfied, by mirroring if necessary, before you make the device unavailable by detaching it.

Detaching a link does not disable sparing. That is, if all links to a physical volume are detached, and a suitable spare physical volume is available in the volume group, LVM

will use it to reconstruct the detached disk. For more information on sparing, see "Increasing Hardware Redundancy Through Disk Sparing" (page 44).

You can view the LVM status of all links to a physical volume using `vgdisplay` with the `-v` option.

Restoring a detached link to a physical volume, or **reattaching** it, makes that link available to the volume group. LVM may begin using that link as necessary to access the disk.

To reattach a specific path to a physical volume, use the `pvchange` command with the `-a` option. For example, enter:

```
# pvchange -a y /dev/dsk/c5t0d0
```

Because detaching a link to a physical volume is meant to be temporary, all detached links in a volume group are reattached when the volume group is activated, either at boot time or with an explicit `vgchange` command, such as:

```
# vgchange -a y /dev/vg02
```

Creating an Alternate Boot Disk

With non-LVM disks, a single root disk contained all the attributes needed for boot up as well as your system files, primary swap, and dump. Using LVM, a single root disk is replaced by a pool of disks, a **root volume group**, which contains all of the same elements but allowing a root logical volume, a boot logical volume, a swap logical volume, and one or more dump logical volumes. Each of these types of logical volumes must be contiguous, that is, contained on a single disk. (Additionally, there can be other noncontiguous logical volumes which might be used for user data.) See *HP-UX System Administrator's Guide: Configuration Management* for more information on the swap and dump devices and their configuration.

The root logical volume contains the operating system software. You have the option of using a separate boot logical volume instead of combining root and boot operations within a single logical volume. Whether you use a single "combined" root-boot logical volume, or separate root and boot logical volumes, the logical volume used to boot the system must be the first logical volume on its physical volume.

If you newly install your HP-UX system and choose the LVM configuration, a root volume group is automatically configured (`/dev/vg00`), as are separate root (`/dev/vg00/lvol3`) and boot (`/dev/vg00/lvol1`) logical volumes. If you currently have a combined root and boot logical volume and you want to reconfigure to separate them, after creating the boot logical volume, you will need to use the `lvlnboot` command with the `-b` option to define the boot logical volume to the system, taking effect the next time the system is booted.

If you decide you want to create a new root volume group that will contain an alternate boot disk, use the following steps.

1. Create a physical volume using `pvcreate` with the `-B` option. The `-B` option creates an area on the disk for a LIF volume, boot utilities, and a BDRA.

NOTE: The BDRA must exist on each bootable disk within the root volume group. The BDRA maintains the information that the kernel requires about the logical volume that contains the root, as well as those that contain primary swap and dump.

See *lif*(4) for more information on LIF volumes.

   a. On an HP Integrity server, partition the disk using the `idisk` command and a partition description file, then run `insf` as described in "Mirroring the Boot Disk on HP Integrity Servers" (page 105).
   b. Run `pvcreate` with the `-B` option. On an HP Integrity server, use the device file denoting the HPUX partition:

      ```
      # pvcreate -B /dev/rdisk/disk6_p2
      ```

      On an HP 9000 server, use the device file for the entire disk:

      ```
      # pvcreate -B /dev/rdisk/disk6
      ```

2. Create a directory for the volume group. For example:

   ```
   # mkdir /dev/vgroot
   ```

3. Create a device file named `group` in the previously described directory with the `mknod` command.

   ```
   # mknod /dev/vgroot/group c 64 0xnn0000
   ```

4. Create the root volume group using the `vgcreate` command, specifying each physical volume to be included:

   ```
   # vgcreate /dev/vgroot /dev/disk/disk6
   ```

5. Use the `mkboot` command to place boot utilities in the boot area:

   ```
   # mkboot /dev/rdisk/disk6
   ```

6. Use `mkboot -a` to add an autoboot file to the disk boot area:

   ```
   # mkboot -a "hpux" /dev/rdisk/disk6
   ```

Now you are ready to create the logical volumes that you intend to use for boot, root, and primary swap. If you create your root volume group with multiple disks, use the

lvextend command to place the boot, root, and primary swap logical volumes on the boot disk (optionally, the primary swap logical volume can be located on a disk other than the boot disk). The boot logical volume must be the first logical volume found on the boot disk. It must begin at physical extent 0000 in order to boot the system in maintenance mode. The boot, root, and primary swap logical volumes must be contiguous (lvcreate option -C y) with bad block relocation disabled (lvcreate option -r n).

TIP: You can use pvmove to move the data from an existing logical volume to another disk, if it is necessary to make room for the root logical volume. See "Moving Data to a Different Physical Volume" (page 88) for more information.

Continue by following these additional steps:

1. Create the boot logical volume. This logical volume will contain the boot file system (/stand). For example, to create a boot logical volume named bootlv with size 512 MB, enter the following commands:

```
# lvcreate -C y -r n -n bootlv /dev/vgroot
# lvextend -L 512 /dev/vgroot/bootlv /dev/disk/disk6
```

2. Create the primary swap logical volume. This logical volume will be the system's primary swap area and will also be used for dump. Optionally, you can configure the primary swap logical volume (and the dump) on a different physical disk than the root logical volume. In this example, the primary swap logical volume is on the same physical disk as the root logical volume. For example, to create a primary swap logical volume named swaplv with size 2 GB, enter the following commands:

```
# lvcreate -C y -r n -n swaplv /dev/vgroot
# lvextend -L 2048 /dev/vgroot/swaplv /dev/disk/disk6
```

3. Create the root logical volume. This logical volume will contain the root file system (/). For example, to create a root logical volume named rootlv with size 1 GB, enter the following commands:

```
# lvcreate -C y -r n -n rootlv /dev/vgroot
# lvextend -L 1024 /dev/vgroot/rootlv /dev/disk/disk6
```

4. Specify that bootlv is the boot logical volume:

```
# lvlnboot -b /dev/vgroot/bootlv
```

5. Specify that rootlv is the root logical volume:

```
# lvlnboot -r /dev/vgroot/rootlv
```

6. Specify that `swaplv` is the primary swap logical volume:

```
# lvlnboot -s /dev/vgroot/swaplv
```

7. Specify that `swaplv` is also to be used for dump:

```
# lvlnboot -d /dev/vgroot/swaplv
```

8. Verify the configuration:

```
# lvlnboot -v /dev/vgroot
Boot Definitions for Volume Group /dev/vgroot:
Physical Volumes belonging in Root Volume Group:
                /dev/disk/disk6 -- Boot Disk
Boot: bootlv    on:     /dev/disk/disk6
Root: rootlv    on:     /dev/disk/disk6
Swap: swaplv    on:     /dev/disk/disk6
Dump: swaplv    on:     /dev/disk/disk6, 0
```

9. Once the boot and root logical volumes are created, you will need to create file systems for them. For example:

```
# mkfs -F hfs /dev/vgroot/rbootlv
# mkfs -F vxfs /dev/vgroot/rrootlv
```

NOTE: On HP Integrity servers, the boot file system can be VxFS:

```
# mkfs -F vxfs /dev/vgroot/rbootlv
```

Mirroring the Boot Disk

NOTE: Mirroring requires the optional product, HP MirrorDisk/UX. Before starting the procedure, be sure that the product is installed. This product is available on the HP-UX 11i application release media. For example:

```
# swlist -l fileset | grep -i mirror
# LVM-MirrorDisk            B.11.31   MirrorDisk/UX
  LVM-MirrorDisk.LVM-MIRROR B.11.31   HP-UX support for the MirrorDisk/UX
```

After you have created mirror copies of the root, boot, and primary swap logical volumes, should any of the underlying physical volumes fail, the system can use the mirror copy on the other disk and continue. When the failed disk comes back online, it will be automatically recovered, provided the system has not been rebooted.

If the system is rebooted before the disk is back online, reactivate the volume group to update the LVM data structures that track the disks within the volume group. You can use `vgchange -a y` even though the volume group is already active.

For example, you can reactivate volume group `vg00` using:

```
# vgchange -a y /dev/vg00
```

As a result, LVM scans and activates all available disks in the volume group, `vg00`, including the disk that came online after the system rebooted.

The procedure for creating a mirror of the boot disk is different for HP 9000 and HP Integrity servers, in that HP Integrity servers use partitioned boot disks.

Mirroring the Boot Disk on HP 9000 Servers

To set up a mirrored root configuration, you must add a disk to the root volume group, mirror all the root logical volumes onto it, and make it bootable. For this example, the disk to be added is at path 0/1/1/0.0x1.0x0 and has device special files named `/dev/rdisk/disk4` and `/dev/disk/disk4`.

1. Use the `insf` command with the `-e` option to make sure the device files are in place. For example:

   ```
   # insf -e -H 0/1/1/0.0x1.0x0
   ```

   You should now have the following device files for this disk:

   `/dev/[r]disk/disk4` *(this refers to the entire disk)*

2. Create a physical volume using `pvcreate` with the `-B` option.

   ```
   # pvcreate -B /dev/rdisk/disk4
   ```

3. Add the physical volume to your existing root volume group with `vgextend`:

   ```
   # vgextend /dev/vg00 /dev/disk/disk4
   ```

4. Use the `mkboot` command to place boot utilities in the boot area:

   ```
   # mkboot /dev/rdisk/disk4
   ```

5. Use the `mkboot` command to add an autoboot file to the disk boot area. If you expect to boot from this disk only when you lose quorum, you can use the alternate string "`hpux -lq`" to disable quorum checking:

   ```
   # mkboot -a "hpux" /dev/rdisk/disk4
   ```

NOTE: This example includes creating a mirror copy of the primary swap logical volume. The primary swap mirror does not need to be on a specific disk or at a specific location, but it must be allocated on contiguous disk space. The recommended mirror policy for primary swap is to have the mirror write cache and the mirror consistency recovery mechanisms disabled.

When primary swap is mirrored and your primary swap device also serves as a dump area, be sure that mirror write cache and mirror consistency recovery is set to off at boot time to avoid loss of your dump. To reset these options, reboot your system in maintenance mode and use the `lvchange` command with the `-M n` and `-c n` options.

6. Use the `lvextend` command to mirror each logical volume in `vg00` (the root volume group) onto the specified physical volume. The logical volumes must be extended in the same order that they are configured on the original boot disk. Use the `pvdisplay` command with the `-v` option to determine the list of logical volumes and their order. For example:

```
# pvdisplay -v /dev/disk/disk0 | grep 'current.*0000 $'
   00000 current /dev/vg00/lvol1 00000
   00038 current /dev/vg00/lvol2 00000
   00550 current /dev/vg00/lvol3 00000
   00583 current /dev/vg00/lvol4 00000
   00608 current /dev/vg00/lvol5 00000
   00611 current /dev/vg00/lvol6 00000
   00923 current /dev/vg00/lvol7 00000
   01252 current /dev/vg00/lvol8 00000
```

In this example, mirror the logical volumes as follows:

```
# lvextend -m 1 /dev/vg00/lvol1 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol2 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol3 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol4 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol5 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol6 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
```

```
# lvextend -m 1 /dev/vg00/lvol7 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol8 /dev/disk/disk4
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
```

7. Update the root volume group information:

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

8. Verify that the mirrored disk is displayed as a boot disk and that the boot, root, and swap logical volumes appear to be on both disks:

   ```
   # lvlnboot -v
   ```

9. Specify the mirror disk as the alternate boot path in nonvolatile memory:

   ```
   # setboot -a 0/1/1/0.0x1.0x0
   ```

10. Add a line to `/stand/bootconf` for the new boot disk using `vi` or another text editor:

    ```
    # vi /stand/bootconf
    l /dev/disk/disk4
    ```

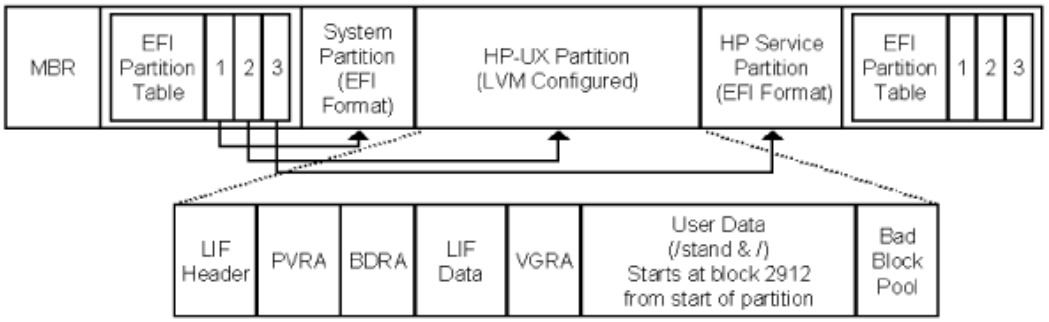
    where the literal "`l`" (lower case L) represents LVM.

Mirroring the Boot Disk on HP Integrity Servers

The procedure to mirror the root disk on Integrity servers is similar to the procedure for HP 9000 servers. The difference is that Integrity server boot disks are partitioned; you must set up the partitions, copy utilities to the EFI partition, and use the HP-UX partition device files for LVM commands.

The following diagram shows the disk layout of a boot disk. The disk contains a Master Boot Record (MBR) and EFI partition tables that point to each of the partitions. The `idisk` command is used to create the partitions (refer to *idisk*(1M)).

Figure 3-1 Example LVM Disk Layout on an HP Integrity Server



For this example, the disk to be added is at hardware path 0/1/1/0.0x1.0x0, with device special files named `/dev/disk/disk2` and `/dev/rdisk/disk2`.

1. Partition the disk using the `idisk` command and a partition description file.
   a. Create a partition description file. For example:

      ```
      # vi /tmp/idf
      ```

      In this example, the partition description file contains:

      ```
      3
      EFI 500MB
      HPUX 100%
      HPSP 400MB
      ```

      NOTE: The values in the example represent a boot disk with three partitions: an EFI partition, an HP-UX partition, and an HPSP. Boot disks of earlier HP Integrity servers might have an EFI partition of only 100 MB and might not contain the HPSP partition.

   b. Partition the disk using `idisk` and your partition description file:

      ```
      # idisk -f /tmp/idf -w /dev/rdisk/disk2
      ```

   c. To verify that your partitions are correctly laid out, run:

      ```
      # idisk /dev/rdisk/disk2
      ```

2. Use the `insf` command with the `-e` option to create the device files for all the partitions. For example:

   ```
   # insf -e -H 0/1/1/0.0x1.0x0
   ```

You should now have the following device files for this disk:

```
/dev/[r]disk/disk2 (this refers to the entire disk)
/dev/[r]disk/disk2_p1(this refers to the efi partition)
/dev/[r]disk/disk2_p2(this will be the hp-ux partition)
/dev/[r]disk/disk2_p3(this refers to the service partition)
```

3. Create a physical volume using `pvcreate` with the `-B` option. Be sure to use the device file denoting the HPUX partition.

   ```
   # pvcreate -B /dev/rdisk/disk2_p2
   ```

4. Add the physical volume to your existing root volume group using `vgextend`:

   ```
   # vgextend vg00 /dev/disk/disk2_p2
   ```

5. Use the `mkboot` command to set up the boot area. Specify the `-e` and `-l` options to copy EFI utilities to the EFI partition, and use the device special file for the entire disk:

   ```
   # mkboot -e -l /dev/rdisk/disk2
   ```

6. Use the `mkboot` command to add an autoboot file to the disk boot area. If you expect to boot from this disk only when you lose quorum, you can use the alternate string "`hpux -lq`" to disable quorum checking:

   ```
   # mkboot -a "hpux" /dev/rdisk/disk2
   ```

7. Use the `lvextend` command to mirror each logical volume in `vg00` (the root volume group) onto the specified physical volume. The logical volumes must be extended in the same order that they are configured on the original boot disk. Use the `pvdisplay` command with the `-v` option to determine the list of logical volumes and their order. For example:

   ```
   # pvdisplay -v /dev/disk/disk0_p2 | grep 'current.*0000 $'
      00000 current /dev/vg00/lvol1 00000
      00010 current /dev/vg00/lvol2 00000
      00138 current /dev/vg00/lvol3 00000
      00151 current /dev/vg00/lvol4 00000
      00158 current /dev/vg00/lvol5 00000
      00159 current /dev/vg00/lvol6 00000
      00271 current /dev/vg00/lvol7 00000
      00408 current /dev/vg00/lvol8 00000
   ```

   In this example, mirror the logical volumes as follows:

   ```
   # lvextend -m 1 /dev/vg00/lvol1 /dev/disk/disk2_p2
   The newly allocated mirrors are now being synchronized.
   ```

```
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol2 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol3 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol4 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol5 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol6 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol7 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
# lvextend -m 1 /dev/vg00/lvol8 /dev/disk/disk2_p2
The newly allocated mirrors are now being synchronized.
This operation will take some time. Please wait ....
```

NOTE: If `lvextend` fails with following message:

```
"m": illegal option
```

then HP MirrorDisk/UX is not installed.

8. Update the root volume group information:

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

9. Verify that the mirrored disk is displayed as a boot disk and that the boot, root, and swap logical volumes appear to be on both disks:

   ```
   # lvlnboot -v
   ```

10. Specify the mirror disk as the alternate boot path in nonvolatile memory:

    ```
    # setboot -a 0/1/1/0.0x1.0x0
    ```

11. Add a line to `/stand/bootconf` for the new boot disk using `vi` or another text editor:

    ```
    # vi /stand/bootconf
    l /dev/disk/disk2_p2
    ```

where the literal "`l`" (lower case L) represents LVM.

Administering File System Logical Volumes

This section describes special actions you must take when working with file systems inside logical volumes. It covers the following topics:

- "Creating a File System" (page 109)
- "Extending a File System" (page 110)
- "Reducing the Size of a File System" (page 112)
- "Backing Up a VxFS Snapshot File System" (page 114)

TIP: When dealing with file systems, you can use HP SMH or a sequence of HP-UX commands. For most tasks, using HP SMH is quicker and simpler. You do not have to explicitly perform each of the following distinct tasks; rather, proceed from the HP SMH Disk and File Systems area. HP SMH will perform all the necessary steps for you.

Creating a File System

When creating either an HFS or VxFS file system in a logical volume, you can use HP SMH or a sequence of HP-UX commands. If you choose to use HP-UX commands directly, the following provides a checklist of subtasks for creating a file system. Most refer to other procedures found elsewhere in this document.

1. If you create a new file system of a type other than HFS, you might need to reconfigure the new type into the kernel. Normally, VxFS will already have been configured into the kernel as part of the default configuration. See *HP-UX System Administrator's Guide: Configuration Management* for information on how to add a file system type.
2. Estimate the size required for the logical volume. To estimate the size requirements for a logical volume containing a file system, see "Setting Up Logical Volumes for File Systems" (page 34).
3. Determine if there is sufficient disk space available in the volume group. Use the `vgdisplay` command to calculate this information, as described in "Creating a Logical Volume" (page 66).

   If there is not enough space within the volume group, you may have to add a disk to a volume group, as described in "Adding a Disk to a Volume Group" (page 64).
4. Create the logical volume. Use `lvcreate`, as described in "Creating a Logical Volume" (page 66).

5. Create the file system using the `newfs` command. Note the use of the character device file. for example:

   ```
   # newfs -F fstype /dev/vg02/rlvol1
   ```

   If you do not use the `-F fstype` option, then `newfs` creates a file system based on the content of your `/etc/fstab` file. If there is no entry for the file system in `/etc/fstab`, then the file system type is determined from the file `/etc/default/fs`. For information on additional options, refer to *newfs*(1M).

   When creating a VxFS file system, file names will automatically be long.

   For HFS, use the `-s` or `-l` option to specify a file system with short or long file names, respectively. By default, the length of file system names are consistent with those of the root file system. Short file names are 14 characters maximum. Long file names allow up to 255 characters. Generally, you use long file names for flexibility; files created on other systems that use long file names can be moved to your system without being renamed.

6. After you have created a file system, you must mount it for users to access it, and add it to `/etc/fstab` so that it will be automatically mounted at boot time.

Extending a File System

Extending a file system inside a logical volume is a two-step task: extending the logical volume, then extending the file system. The first step is described in "Extending a Logical Volume" (page 67). The second step, extending the file system itself, depends on several factors:

- What type of file system is involved? If it is HFS or VxFS? HFS requires the file system to be unmounted to be extended.

  Check the type of file system using the `fstyp` command. For example:

  ```
  # /usr/sbin/fstyp /dev/vg01/lvol2
  vxfs
  ```

- If the file system is VxFS, do you have the base VxFS product or the OnlineJFS product? If you have only the base VxFS product, you must unmount the file system before extending it.

  To see if the OnlineJFS product is installed, use `swlist`, checking for the `OnlineJFS` product:

  ```
  # swlist -l product | grep -i OnlineJFS
    OnlineJFS  B.11.31   Online features of the VxFS File System
  ```

- Can you unmount the file system? To unmount system directories such as `/var` and `/usr`, you must be in single-user mode.
- Is the file system the root file system (`/`)? If so, there are two complications:
  — The logical volume containing the root file system is created with the contiguous allocation policy, so it may not be possible to extend it in place.
  — The root file system cannot ever be unmounted, even if you shut down to single-user state.

  If you are using VxFS as your root file system and have the OnlineJFS product, you can extend the original root file system without unmounting, provided there is contiguous disk space available.

  Otherwise, to extend the current root file system, you must have created and mounted *another* root disk which enables you to work with the unmounted original root disk, extending it *if* there is contiguous disk space still available. If the original disk does not have contiguous disk space available, instead of expanding the original root disk, you can create a new root file system on another larger disk.

Once you have the answers to these questions, you can follow this procedure:

CAUTION: This procedure may fail on a VxFS file system already at 100% capacity (Error 28). You must remove some files before you attempt this operation.

1. If the file system must be unmounted, unmount it.
   a. Be sure no one has files open in any file system mounted to this logical volume and that it is no one's current working directory. For example:

      ```
      # fuser -cu /work/project5
      ```

      If the logical volume is in use, confirm that the underlying applications no longer need it. You will likely have to stop the applications.

      NOTE: If the file system is exported via NFS to other systems, verify that no one is using those other systems, and then unmount it on those systems.

   b. If you cannot stop the applications using the logical volume, or it is a system directory like `/var` and `/usr`, change to single-user state:

      ```
      # /sbin/shutdown
      ```

   c. Unmount the file system:

      ```
      # /sbin/umount /dev/vg01/lvol2
      ```

2. Extend the logical volume. For example:

   ```
   # /sbin/lvextend -L 332 /dev/vg01/lvol2
   ```

increases the size of this volume to 332 MB.

3. Extend the file system size to the logical volume size. If the file system is unmounted, use the `extendfs` command:

   ```
   # /sbin/extendfs /dev/vg01/rlvol2
   ```

   If you did not have to unmount the file system, use the `fsadm` command instead. The new size is specified in terms of the block size of the file system. In this example, the block size of the file system `/work/project5` is 1 KB. To extend the file system to 332 MB, the number of blocks is 339968 (332 times 1024):

   ```
   # fsadm -b 339968 /work/project5
   ```

4. If you had to unmount the file system, mount it again.
   a. If you had to change to single-user state, reboot the system.

      ```
      # /sbin/reboot -r
      ```

      You can skip any additional steps to mount the file system and export it, since the boot process should mount and export any file systems.

   b. Remount the file system:

      ```
      # /sbin/mount /dev/vg01/rlvol2 /mount_point
      ```

NOTE: If the file system will continue to be used by NFS clients, export it on the server (`exportfs -a`) and remount it on the clients (`mount -a`).

Upon completing these steps, verify that the file system reflects the expansion by entering `bdf`, `df`, or `fsadm -E`.

Reducing the Size of a File System

You might want to shrink a file system that has been allocated more disk space than will be needed, allowing that disk space to be freed for some other use.

Reducing the size of a file system is more complicated than extending it. Because of file system block allocation strategies, data may be scattered throughout the logical volume; reducing the logical volume will reclaim space at the end of the logical volume, requiring file system drivers to coalesce and rearrange data blocks ahead of time. Most types of file system are unable to do such coalescence, so you must back up the data in the file system, reduce the logical volume, create a new file system in the smaller logical volume, and restore the data from your backup.

The only current file system type able to do online coalescence and size reduction is OnlineJFS, and it may fail in some cases.

Reducing a File System Created with OnlineJFS

Using the `fsadm` command shrinks the file system, provided the blocks it attempts to deallocate are not currently in use; otherwise, it will fail. If sufficient free space is currently unavailable, file system defragmentation of both directories and extents, previously described, might enable you to consolidate free space toward the end of the file system, allowing the contraction process to succeed when subsequently retried.

For example, suppose your VxFS file system is currently 6 GB. However, you decide you really only need 2GB with an additional 1 GB for reserve space. As a result, you want to resize the file system to a new size of 3 GB. Use `fsadm` with the `-b` option to specify the new size of the file system in sectors, then reduce the size of the logical volume to match. Assuming the file system sector size is 1K, use the following commands:

```
# fsadm -b 3145728 /home
# lvreduce -L 3072 /dev/vg01/lvol5
```

Reducing a File System Created with HFS or VxFS

1. Be sure no one has files open in any file system on the logical volume and that the logical volume is no one's current working directory:

   ```
   # fuser -cu /dev/vg01/lvol5
   ```

NOTE: If the file system is exported via NFS to other systems, verify that no one is using those other systems, and then unmount the file system on those systems before unmounting it on the server.

2. Back up the data in the logical volume.

   For example, to back up `/work/project5` to the system default tape device:

   ```
   # tar cv /work/project5
   ```

3. Remove the data in the file system the logical volume is mounted to:

   ```
   # rm -r /work/project5
   ```

   Because `/work/project5` is a mount point, `rm -r` will not remove the directory itself.

4. Unmount the file system to which the logical volume is mounted:

   ```
   # umount /work/project5
   ```

5. Reduce the size of the logical volume:

   ```
   # lvreduce -L 500 /dev/vg01/lvol5
   ```

This command reduces the logical volume `/dev/vg01/lvol5` to 500 MB.

6. Create a new file system in the reduced logical volume, using the `newfs` command. Note the use of the character device file. For example:

   ```
   # newfs -f fstype /dev/vg01/rlvol5
   ```

7. Mount the logical volume:

   ```
   # mount /dev/vg01/lvol5 /work/project5
   ```

8. Recover the data from the backup; for example,

   ```
   # tar xv
   ```

   Recovers all the contents of a tape in the system default drive.

9. If `/work/project5` will continue to be used by NFS clients, re-export it on the server (`exportfs -a`) and remount it on the clients (`mount -a`).

Backing Up a VxFS Snapshot File System

NOTE: Creating and backing up a VxFS snapshot file system requires that you have the optional HP OnlineJFS product installed on your system. See *HP-UX System Administrator's Guide: Configuration Management* for more information.

VxFS enables you to perform backups without taking the file system offline by making a snapshot of the file system, a read-only image of the file system at a moment in time. The primary file system remains online and continues to change. After you create the snapshot, you can back it up with any backup utility except the `dump` command.

1. Determine how large the snapshot file system must be, and create a logical volume to contain it. Use `bdf` to assess the primary file system size and consider the following:
   - Block size of the file system (1,024 bytes per block by default)
   - How much the data in this file system is likely to change (HP recommends 15 to 20% of total file system size)

   For example, to determine how large to make a snapshot of `lvol4`, mounted on the `/home` directory, examine its `bdf` output:

   ```
   # bdf /home
   filesystem          kbytes    used   avail %used mounted on
   /dev/vg00/lvol4      40960   38121    2400   94% /home
   ```

Allowing for 20% change to this 40 MB file system, you would want to create a logical volume of eight blocks (8 MB).

2. Use `lvcreate` to create a logical volume to contain the snapshot file system.

   For example,

   ```
   # lvcreate -L 8 -n snap /dev/vg02
   ```

   creates an 8 MB logical volume called `/dev/vg02/snap`, which should be sufficient to contain a snapshot file system of `lvol4`.

   Refer to *lvcreate*(1M) for syntax.

3. Create a directory for the mount point of the snapshot file system.

   For example,

   ```
   # mkdir /tmp/house
   ```

4. Create and mount the snapshot file system.

   In the following example, a snapshot is taken of logical volume `/dev/vg00/lvol4`, contained in logical volume `/dev/vg02/snap`, and mounted on `/tmp/house`:

   ```
   # mount -f vxfs -o snapof=/dev/vg00/lvol4 /dev/vg02/snap /tmp/house
   ```

   Refer to *mount_vxfs*(1M) for syntax.

5. Back up the snapshot file system with any backup utility except `dump`.

   For example, to use *tar*(1M) to archive the snapshot file system `/tmp/house`, ensuring that the files on the tape have relative path names:

   ```
   # cd tmp; tar cf /dev/rtape/tape0BEST house
   ```

   Alternatively, the following *vxdump*(1M) command backs up a snapshot file system `/tmp/house`, which has extent attributes:

   ```
   # vxdump -0 -f /dev/rtape/tape0BEST /tmp/house
   ```

Administering Swap Logical Volumes

When you enable a swap area within a logical volume, HP-UX determines how large the area is, and it will use no more space than that. If your disk has enough remaining contiguous space, you can subsequently increase the size of your primary swap area by using the `lvextend` command (or HP SMH) to enlarge the logical volume and then reboot the system. This procedure allows HP-UX to use the extra space that you have provided.

If you plan device swap areas in addition to primary swap, you will attain the best performance when the device swap areas are on different physical volumes. This configuration allows for the interleaving of I/O to the physical volumes when swapping occurs.

To create interleaved swap, create multiple logical volumes for swap, with each logical volume on a separate disk. You must use HP-UX commands to help you obtain this configuration; HP SMH does not allow you to create a logical volume on a specific disk. See "Extending a Logical Volume to a Specific Disk" (page 68).

You can configure your swap space as described in *HP-UX System Administrator's Guide: Overview*.

NOTE: You must reboot the system for the system to recognize changes to the swap configuration.

Extending a Swap Device

If you are using a logical volume for swap, you must increase the logical volume size *before* increasing the swap size. You can extend the logical volume using `lvextend` or HP SMH.

Note that swap logical volumes must be contiguous, so extending the logical volume will succeed only if there are physical extents available at the end of the existing logical volume. If contiguous disk space is not available, create a new contiguous logical volume for primary swap within the root volume group. You do not need to designate a specific disk. For example:

```
# lvcreate -C y -L 48 -n pswap /dev/vgroot
```

After creating a logical volume that will be used as primary swap, use `lvlnboot` to update the boot information:

```
# lvlnboot -s /dev/vgroot/pswap
```

Reducing the Size of a Swap Device

If you are using a logical volume for swap, you must reduce the swap size *before* reducing the size of the logical volume. You can reduce the size of the logical volume using `lvreduce` or HP SMH.

Hardware Issues

This section describes hardware-specific issues dealing with LVM.

Integrating Cloned LUNs Using the vgchgid Command

Certain disk arrays have the ability to create clones of its LUNs. For example, the HP XP product enables the system administrator to split off a set of LUNs, called Business Copies (BCs), which are copies of existing LUNs.

Cloned disks have the same information in their LVM headers as the original disks, which violates LVM's requirement that each disk have a unique identifier. To make the cloned disks usable with LVM, the `vgchgid` command can be used to change their volume group identifier (VGID).

All of the physical volumes to be changed must belong to the same volume group. Therefore, if you are changing multiple physical volumes, specify all of them in a single invocation of `vgchgid`. Otherwise, they will be assigned different VGIDs.

For example, suppose you have a volume group containing four physical volumes and create a BC for each physical volume. If you run `vgchgid` on only two BCs, `vgchgid` modifies the VGID on those two BCs. If you then run `vgchgid` again with all four BCs, `vgchgid` reports that they belong to different volume groups. To correct this, you can either run `vgchgid` on the two unmodified BCs and then use the four BCs in two separate volume groups, or you can merge back the two modified BCs and split them off again before finally running `vgchgid` with all four BCs.

After executing `vgchgid` on a set of physical volumes, use `vgimport` to import them into a new volume group. An example showing how `vgchgid` might be used follows:

1. Make BC copies and create new device files using the instructions for the array.
2. Change the VGID on the cloned disks:

   ```
   # vgchgid /dev/rdisk/disk49 /dev/rdisk/disk50
   ```

3. Create the volume group directory and `group` file:

   ```
   # mkdir /dev/vg04
   # mknod /dev/vg04/group c 64 0x40000
   ```

4. Import the physical volumes:

   ```
   # vgimport /dev/vg04 /dev/rdisk/disk49 /dev/rdisk/disk50
   ```

5. Back up the volume group configuration information:

   ```
   # vgcfgbackup /dev/vg04
   ```

6. Activate the volume group:

   ```
   # vgchange -a y /dev/vg04
   ```

4 Troubleshooting LVM

This chapter contains the following information:

LVM Concepts

This section contains information about the inner workings of LVM. It may prove useful when tracking down system errors or interpreting error messages. This information is subject to change.

Characteristics and Layout of LVM Disks

There are two kinds of LVM disk layouts—for boot disks and all other LVM disks—and they differ in their data structures. Non-bootable disks have two reserved areas: the physical volume reserved area (PVRA) and the volume group reserved area (VGRA). Bootable disks have a PVRA and VGRA, and additional sectors reserved for the boot data reserved area (BDRA) and boot LIF.

Boot Data Reserved Area (BDRA)

The boot data reserved area (BDRA) contains the information needed to configure the root, primary swap, and dump logical volumes, and to mount the root file system.

Information about the LVM disk data structures in the BDRA is maintained by using the `lvlnboot` and `lvrmboot` commands. Here is sample output, followed by explanation:

```
# lvlnboot -v
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
                /dev/dsk/c3t0d0 -- Boot Disk
                /dev/dsk/c4t0d0 -- Boot Disk
                /dev/dsk/c5t0d0
                /dev/dsk/c12t0d0 -- Boot Disk
Root: lvol1     on: /dev/dsk/c3t0d0
                    /dev/dsk/c4t0d0
Swap: lvol2     on: /dev/dsk/c3t0d0
```

```
                    /dev/dsk/c4t0d0
Dump: lvol2     on: /dev/dsk/c3t0d0
```

The physical volumes designated "Boot Disk" are bootable, having been initialized with `mkboot` and `pvcreate -B`. Multiple lines for `lvol1` and `lvol2` indicate that the root and swap logical volumes are being mirrored.

Logical Interface Format area (LIF)

LVM boot disks contain a LIF area, in which is stored a `LABEL` file. On HP 9000 servers, the LIF area contains boot utilities such as the initial system loader (`ISL`), the kernel boot loader (`HPUX`), and the autoboot file (`AUTO`), as well as offline diagnostics.

The `LABEL` file is created and maintained by `lvlnboot` and `lvrmboot`. It contains information about the starting point and size of boot-relevant logical volumes, including the boot file system (`/stand`). Utilities can use the `LABEL` file to access the root, primary swap, and dump logical volumes without actually using LVM.

Physical Volume Reserved Area (PVRA)

The physical volume reserved area (PVRA) contains information describing the physical volume, such as its unique identifier and physical extent information, as well as pointers to other LVM structures on the disk.

Volume Group Reserved Area (VGRA)

The volume group reserved area (VGRA) describes the volume group to which the disk belongs. The information is replicated on all of the physical volumes and updated whenever a configuration change is made. Among other data, it contains:

- A list of physical volumes in the volume group, including physical volume status and size, and a map of physical extents to logical volumes.
- A list of logical volumes in the volume group, including the status and capabilities of each logical volume, its scheduling and allocation policies, and the number of mirror copies.
- A volume group header containing the volume group identifier (VGID) and three configurable parameters: the number of physical volumes allowed in the volume group, the maximum number of logical volumes allowed in the volume group, and the maximum number of physical extents allowed per physical volume.

Since each physical extent is recorded in the VGRA, the extent size has a direct bearing on the size of the VGRA. In most cases, the default extent size will work perfectly well. However, if you run into problems, you might consider that the VGRA is a fixed size and a high-capacity physical volume might exceed the total number of physical extents allowed. As a result, you might need to use a larger-than-default extent size on high-capacity LVM disks. Conversely, if all LVM disks in a volume group are small, the default number of extents might make the VGRA too large, wasting disk and memory space. A smaller-than-default extent size or number of physical extents might

be preferable. A high-capacity physical volume might be unusable in a volume group whose extent size is small or set with a small number of physical extents per disk.

User Data Area

The user data area is the region of the LVM disk used to store all user data, including file systems, virtual memory system (swap), or user applications.

Device Number Format

The device files associated with LVM reside in the `/dev` directory. For each volume group, there is a directory under `/dev`, named after the volume group; in that directory is a single "group" device file and separate block and character device files for each logical volume.

Here is a sample listing:

```
# ls -l /dev/vg01
total 0
crw-r--r--   1 root    root     64 0x010000 Mar 28  2004 group
brw-r-----   1 root    root     64 0x010001 Jul 29 16:53 lvol1
brw-r-----   1 root    root     64 0x010002 Jul 29 16:53 lvol2
crw-r-----   1 root    root     64 0x010001 Mar 28  2004 rlvol1
crw-r-----   1 root    root     64 0x010002 Mar 28  2004 rlvol2
```

The format of the device file number is shown in the table below:

Table 4-1 Logical Volume Manager Device Number Format

Major Number (8 bits)	Volume Group Number (8 bits)	Reserved (8 bits)	Logical Volume Number (8 bits)
64	0–0xff	0	0–0xff 0=group file

The major number for all LVM device files is 64. The volume group number is encoded into the top eight bits of the minor number, and the logical volume number is encoded into the low eight bits. Logical volume number 0 is reserved for the "group" file.

By default, volume group numbering begins with zero (`vg00`), while logical volumes begin with one (`lvol1`). This is because the logical volume number corresponds to the minor number and the volume group's group file is assigned minor number 0.

Physical volumes use the device files associated with their disk. LVM does not create device files for physical volumes.

Troubleshooting Tools Overview

Information Collection

You can collect information about your LVM configuration using the `vgdisplay`, `lvdisplay`, `pvdisplay`, and `lvlnboot` commands. As noted in "Configuring for Optimal Recovery" (page 52), you should periodically collect the outputs from the following commands:

Table 4-2 LVM Information to Collect and Maintain

Command	Scope	Purpose
`ioscan -f`		Print I/O configuration
`lvlnboot -v`		Print information on root, boot, swap, and dump logical volumes
`vgcfgrestore -l`	for all volume groups	Print volume group configuration from backup file
`vgdisplay -v`	for all volume groups	Print volume group information, including status of logical volumes and physical volumes
`lvdisplay -v`	for all logical volumes	Print logical volume information, including mapping and status of logical extents
`pvdisplay -v`	for all physical volumes	Print physical volume information, including status of physical extents

Consistency Checks

Most LVM commands perform consistency checking. You can inspect your LVM configuration with the `vgdisplay`, `lvdisplay`, and `pvdisplay` commands, and look for inconsistencies.

In addition, there is one command to perform explicit consistency checking on a physical volume: `pvck`. This command detects bad checksums caused by a forward system migration after a backward system migration and should only be run on a deactivated volume group. See *pvck*(1M) for more information.

Maintenance Mode Boot (Booting Without LVM)

LVM **maintenance mode boot** is a special way to boot your system that bypasses the normal LVM structures. It should be used only for problems that prevent the system from otherwise booting. It is similar to single-user state in that many of the processes that normally get started are not started, nor are many of the system checks that are normally performed. It is intended to enable you to boot your system long enough for

you to repair damage to the system LVM data structures typically using `vgcfgrestore`, which should then enable you to boot your system normally.

Normally, the boot loader uses the `LABEL` file in the LIF volume to determine the location of the boot file system (`/stand`) and the kernel `/stand/vmunix`. The `LABEL` file also contains the starting block and size of the root file system (`/`).

Under a maintenance mode boot, the boot loader attempts to find the boot file system at the start of the boot disk's user data area, rather than using information from the LIF volume. To obtain the root file system's starting block and size, the boot loader reads the file `/stand/rootconf`. Since LVM is not enabled, the root file system must be allocated contiguously.

A maintenance mode boot differs from a standard boot in the following ways:

- The system is booted in single-user mode.
- No volume groups are activated.
- Primary swap and dump are not available.
- Only the root file system (`/`) and boot file system (`/stand`) are available.
- If the root file system is mirrored, only one copy is used. Changes to the root file system are not propagated to the mirror copies, but those mirror copies are marked stale and will be synchronized when the system is booted normally.

To boot in maintenance mode on a system with a root disk configured with LVM, use the `-lm` option to the boot loader. On an HP 9000 server, enter this command:

```
ISL> hpux -lm
```

On an HP Integrity server, enter this command:

```
HPUX> boot -lm
```

CAUTION: When you boot your system in maintenance mode, do not activate the root volume group and do not change to multi-user mode (for example, by specifying `/sbin/init 2`). Doing so could corrupt the root file system.

When you have repaired or restored the LVM configuration information, reboot the system:

```
# /usr/sbin/reboot
```

Further information about LVM maintenance mode boots and troubleshooting problems with LVM structures can be found in *Disk and File Management Tasks on HP-UX*, published by Prentice Hall PTR, 1997.

Log Files and Trace Files

LVM does not have a dedicated log file or trace file. It logs any errors or warnings to `/var/adm/syslog/syslog.log`.

I/O Errors

When a device driver returns an error to LVM on an I/O request, LVM classifies the error as either **recoverable** or **non-recoverable**. How those errors are handled determines your course of action.

Recoverable Errors

When LVM encounters a recoverable or correctable error, it internally retries the failed operation under the assumption that the error will correct itself or that you as system administrator can take steps to correct it. Examples of recoverable errors are device power failure, a disk that goes missing *after the volume group is activated*, or a loose disk cable—which can manifest itself as a missing disk. In these cases, LVM logs an error message to the console, but it does not return an error to the application accessing the logical volume.

If you have a current copy of the data on a separate, functioning mirror, then LVM directs the I/O to a mirror copy, much as it would for a non-recoverable error. Applications accessing the logical volume will not detect any error. (To preserve data synchronization between its mirrors, LVM retries recoverable write requests to a problematic disk, even if there is a current copy elsewhere; however, this is managed by a daemon internal to LVM and has no impact on user access to the logical volume.)

If, however, the device in question holds the only copy of the data, LVM retries the I/O request until it succeeds—that is, until the device responds or the system is rebooted. Any application performing I/O to the logical volume might block, waiting for the device to recover. In this case, your application or file system might appear to be stalled and might be unresponsive.

Temporarily Unavailable Device

By default, LVM retries I/O requests with recoverable errors until they succeed or the system is rebooted. Therefore, if an application or file system stalls, your troubleshooting should include checking the console log for problems with your disk drives and taking action to restore the failing devices to service.

Permanently Unavailable Device

If, for some reason, retrying the I/O request will never succeed—such as the disk was physically removed—your application or file system might block indefinitely. If your application is not responding, you might have to reboot your system.

As an alternative to rebooting, you can control how long LVM retries a recoverable error before treating it as non-recoverable by setting a timeout on the logical volume.

Logical Volume Timeouts

You can set the maximum length of time that LVM retries an I/O request using the `-t` option of the `lvchange` command. This sets the timeout value in seconds for a logical volume. For example, to set the timeout for `/dev/vg01/lvol1` to one minute, enter:

```
# lvchange -t 60 /dev/vg01/lvol1
```

This command sets the maximum length of time that LVM retries an I/O request. If the device fails to respond within that time, LVM returns an I/O error to the caller. The timeout value is normally zero, which is interpreted as an infinite timeout; thus, by default, no I/O request returns to the caller until it completes successfully. This timeout value is handled as a best effort, so it could be that the I/O error is returned seconds after the logical volume timeout expired.

If you want to enable a timeout on a logical volume, set it to an integral multiple of any timeout assigned to the underlying physical volumes. Otherwise, the actual duration of the I/O request might exceed the logical volumes timeout. See *pvchange*(1M) for details on how to change the I/O timeout value on a physical volume.

You can view the timeout value for a logical volume using the `lvdisplay` command.

CAUTION: Setting a timeout on a logical volume increases the likelihood of transient errors being treated as non-recoverable errors, so any application that reads or writes to the logical volume might experience I/O errors. If your application is not prepared to handle such errors, keep an infinite logical volume timeout.

Non-Recoverable Errors

Non-recoverable errors are considered fatal; there is no expectation that retrying the operation could work.

If you have a current copy of the data on a separate, functioning mirror, then LVM directs reads and writes to a mirror copy. As far as the application accessing the logical volume is concerned, the I/O operation completes successfully.

However, if you have no other copies of the data—that is, the only copy of the data is on that physical volume—then LVM returns an error to whatever subsystem is accessing the logical volume. This means that any application directly accessing a logical volume should be prepared for I/O requests to fail. File systems such as VxFS and most database applications are designed to recover from error situations; for example, if VxFS encounters an I/O error, it might disable access to a file system or a subset of the files in it.

LVM considers the following two, specific situations as non-recoverable. How you deal with the error depends on what kind of problem LVM encountered.

Media Errors

If an I/O request fails because of a media error, LVM typically prints a message to the console log file (`/var/adm/syslog/syslog.log`) when the error occurs. In the event of a media error, you must replace the disk (see "Replacing a Bad Disk" (page 131)).

If your disk hardware supports automatic bad block relocation (usually known as "hardware sparing"), enable it, as it will minimize media errors seen by LVM.

NOTE: At one time, LVM performed bad block relocation in software, but now defers to the hardware bad block relocation implemented within modern disks and disk arrays. LVM recognizes and honors software relocation entries created by previous releases but will not create new ones. Enabling or disabling bad block relocation using `lvchange` has no effect.

Missing Device When the Volume Group Was Activated

If the device associated with the I/O was not present *when the volume group was activated*, LVM prints an error message to the user's terminal at activation time. You must either locate the disk and restore it to service, or replace it, then activate the volume group again.

Volume Group Activation Failures

Normally, volume groups are automatically activated during system startup. Unless you intentionally deactivate a volume group using `vgchange`, you will probably not need to activate a volume group. In all cases, LVM requires that a quorum of disks in a volume group be available.

Quorum is the required number of physical volumes that must be available in a volume group in order to activate that volume group or for it to remain activated. To activate a volume group, *more than half* its disks that were available during the last activation must be online and in service; for the volume group to remain fully operational, *at least half* the disks must remain present and available.

During run time, when a volume group is already active, if a disk fails or is taken offline, the quorum might become lost. This condition occurs if less than half of the physical volumes defined for the volume group now remain fully operational. For example, if there are two disks in the volume group, the loss of one would not cause a loss of quorum, as is the case when activating the volume group; rather, both disks would need to become unavailable. If this happens, your volume group remains active; however, a message is printed to the console, indicating that the volume group has lost quorum. Until the quorum is restored (at least one of the LVM disks in the volume group in the previous example is again available), LVM will not allow you to complete most commands that affect the volume group configuration. Further, some of the I/O accesses to the logical volumes for that volume group might hang because the underlying disks are not accessible. Also, until quorum is restored, the MWC will not

be updated because LVM cannot guarantee the consistency (integrity) of the LVM information.

The `vgchange -q n` option can be used to override the system's quorum check. Overriding quorum can result in a volume group whose configuration is inaccurate (for example, missing recently creating logical volumes). This configuration change might not be reversible.

There are ways to override quorum requirements at volume group activation time or boot time. Even when allowed by LVM, HP recommends that you do not make changes to the LVM configuration for active volume groups that do not have a quorum of disks present. To correct quorum issues, HP recommends returning the unavailable disks to service.

Quorum Problems with a Non-Root Volume Group

If you attempt to activate a non-root volume group when not enough disks are present to establish a quorum, you will see error messages similar to the following:

```
# vgchange -a y /dev/vg01
vgchange: Warning: Couldn't attach to the volume group
                   physical volume "/dev/dsk/c1t0d2":
The path of the physical volume refers to a device that does not exist,
      or is not configured into the kernel.
vgchange: Couldn't activate volume group "/dev/vg01":
Either no physical volumes are attached or no valid VGDAs were found on
    the physical volumes.
```

If a non-root volume group does not get activated because of a failure to meet quorum:

1. Check the power and data connections (including Fibre Channel zoning and security) of all the disks that are part of the volume group that you cannot activate. Return all disks (or at least enough to make a quorum) to service. Then use the `vgchange` command to activate the volume group again.
2. If there is no other way to make a quorum available, use the `-q` option of the `vgchange` command to override the quorum requirement.

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

   As a result, the volume group activates without a quorum being present. You might get messages about not being able to access certain logical volumes because part or all of a logical volume might be located on one of the disks that is not present.

   Whenever you override a quorum requirement, you run the risk of using data that are not current. Be sure to check the data on the logical volumes in the activated volume group, as well as the size and locations of the logical volumes, to ensure that they are up to date.

Return the disabled disks to the volume group as soon as possible. When you return a disk to service that was not online when you originally activated the volume group, again use the `vgchange` command:

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

Quorum Problems with Your Root Volume Group

Your root volume group might also have a quorum problem. If there are not enough disks present in the root volume group to constitute a quorum, a message indicating that not enough physical volumes are present is displayed during the boot sequence. This error might occur if you have physically removed a disk from your system because you no longer intended to use it with the system but did not remove the physical volume from the volume group using `vgreduce`. Although you should never remove an LVM disk from a system without first removing it from its volume group, you can probably recover from this situation by booting your system with the quorum override option, `hpux -lq`.

Root Volume Group Scanning

If the LVM subsystem detects that some vital information is corrupted on the boot disk, it will scan all the attached devices to try to find the physical volumes that are part of the root volume group. You will then see the following messages on the system console and in `/var/adm/syslog/syslog.log`:

```
LVM : Failure in attaching PV (dev=0x10000nn) to the root volume group.
The physical volume does not belong to the root volume group
LVM : Failure in attaching PV (dev=0x10000nn) to the root volume group.
The physical volume does not belong to the root volume group
LVM : Activation of root volume group failed
Quorum not present, or some physical volume(s) are missing
LVM: Scanning for Root VG PVs (VGID 0xnnnnnnnn 0xnnnnnnnn)
```

If this root volume group scanning succeeds, you will see messages similar to:

```
LVM: Rootvgscan detected 10 PV(s).  Will attempt root VG activation
using the following PV(s):
        0x100005f 0x1000060 0x1000061 0x1000062 0x1000063 0x1000064
        0x1000065 0x1000067 0x1000068 0x100006e
LVM: WARNING: Root VG activation required a scan.  The PV information in
        the on-disk BDRA may be out-of-date from the system's current IO
        configuration.  To update the on-disk BDRA, first update /etc/lvmtab
        using vgscan(1M), then update the on-disk BDRA using lvlnboot(1M).
        For example, if the root VG name is /dev/vg00:
                1. vgscan -k -f /dev/vg00
                2. lvlnboot -R /dev/vg00
LVM: Root VG activated
```

If this root volume group scanning fails to find all physical volumes, you will see the following message:

```
LVM: WARNING: Rootvgscan did not find any PV(s) matching root VGID.
Will attempt root VG activation using the boot device (0x10000nn).
```

Or:

```
LVM: WARNING: BDRA lists the number of PV(s) for the root VG as nn,
but rootvgscan found only nn.  Proceeding with root VG activation.
```

LVM Boot Failures

There are several reasons why an LVM configuration cannot boot. In addition to the same kinds of problems associated with boots from non-LVM disks, the following could cause an LVM-based system not to boot:

Insufficient Quorum

Under this scenario, not enough disks are present in the root volume group to meet the **quorum** requirements. At boot time, you see a message indicating that not enough physical volumes are available:

```
panic: LVM: Configuration failure
```

To activate the root volume group and successfully boot the system, the number of available LVM disks must be more than half the number of LVM disks that were attached when the volume group was last active. Thus, if during the last activation there were two disks attached in the root volume group, the "more than half" requirement means that both must be available. See "Volume Group Activation Failures" (page 126) for information on how to deal with quorum failures.

Corrupted LVM Data Structures on Disk

The LVM bootable disks contain vital boot information in the BDRA. This information might have become corrupted, not current, or just no longer present. Because of the importance of maintaining up-to-date information within the BDRA, use the `lvrmboot` and/or `lvlnboot` commands whenever you make a change that affects the location of the root, boot, primary swap, or dump logical volumes.

Correcting such problems require booting the system in maintenance mode, as described in "Maintenance Mode Boot (Booting Without LVM)" (page 122), and repairing the damage to the system LVM data structures. Typically this can be done by using `vgcfgrestore` on the boot disk.

Corrupted LVM Configuration File

Another possible problem pertaining to activation of a volume group is a missing or corrupted `/etc/lvmtab` file. After booting in maintenance mode, you can use the `vgscan` command to re-create the `/etc/lvmtab` file. See *vgscan*(1M) for more information.

Problems After Reducing the Size of a Logical Volume

When a file system is first created within a logical volume, it is made as large as the logical volume will permit.

If you extend the logical volume without extending its file system, you can subsequently safely reduce the logical volume size, as long as it remains as big as its file system. (Use *bdf*(1M) to determine the size of your file system.) After you expand the file system, you can no longer safely reduce the size of the associated logical volume.

If you reduce the size of a logical volume containing a file system to a size smaller than that of a file system within it using the `lvreduce` command, you will corrupt the file system. If you subsequently attempt to mount the corrupt file system, you might crash your system. If this occurs:

1. Reboot your system in single-user state.
2. If you do not have such backup data and if that data is critical, try to recover whatever part of the data that might remain intact by attempting to back up the files on that file system in your usual way. If you already have a good current backup of the data in the now corrupt file system, omit this step.

   Before you attempt any current backup, be aware that:

   - When your backup program accesses the corrupt part of the file system, your system will crash again. You will need to reboot your system again to continue with the next step.
   - There is no guarantee that all (or any) of your data on that file system will be intact or recoverable. This step is an attempt to save as much as possible. That is, any data successfully backed up in this step will be recoverable, but some or all of your data might not allow for successful backup because of file corruption.
3. Immediately unmount the corrupted file system if it is mounted.
4. Use the logical volume for swap space or raw data storage, or use HP SMH or the `newfs` command to create a new file system in the logical volume. This new file system will now match the current reduced size of the logical volume.
5. If you have created a new file system on the logical volume, do one of the following:
   - If you have a good prior backup (not the backup from step 2), restore its contents. Because the new file system in the smaller logical volume will be

smaller than the original file system, you might not have enough space to restore all your original files.

- If you do not have a good prior backup, attempt to restore as many files as possible from any backup you made in step 2. Again, there is no guarantee that complete data will be recoverable from this backup.
- Use the new file system for creating and storing a new set of files (not for trying to restore the original files).

Replacing a Bad Disk

Since disks are physical devices, their hardware can fail, necessitating their replacement. After a failing disk is replaced with a new one (retaining the hardware address of the original to avoid confusion), the data must be restored to that disk from a backup.

Since the disk was under LVM control, it could have physical extents for several logical volumes on it. The layout of those logical volumes must first be restored and the data for each of those logical volumes restored from backup.

This section provides a step-by-step guide to replacing a faulty LVM disk and outlines the general sequence of commands required to perform the task.

Review "Preparing for the Recovery of LVM System" (page 53) for steps that should be performed before a disk fails. Be sure that you read this section carefully, and implement the required procedures as soon as possible. Your system recovery might rely on these steps. HP recommends that you familiarize yourself with the procedures outlined in this document before you need them so that you understand fully the steps involved.

If you have any questions about the recovery process, contact your local HP Customer Response Center for assistance.

TIP: For an in-depth discussion of disk failures, see the white paper *When Good Disks Go Bad: Dealing with Disk Failures under LVM*, available at http://docs.hp.com. It covers additional topics such as recognizing a disk failure, identifying the failing disk, and choosing the appropriate resolution, such as removing the disk instead of replacing it. The paper also covers releases prior to HP-UX 11i Version 3. The procedures below are summarized from the white paper.

Before Replacing a Disk

Once you have isolated a failed disk, the replacement process depends on answers to the following questions:

- **Is the disk hot-swappable?**

  "Hot-swappable" implies the ability to remove or add an inactive hard disk drive module to a system while power is still on and the SCSI bus is still active. In other

words, you can replace or remove a hot-swappable disk from a system without turning off the power to the entire system.

If your disk is not hot-swappable, you will have to schedule system down time to replace the disk.

Consult your system hardware manuals for information about which disks in your system are hot-swappable. Specifications for other hard disks are available in their installation manuals at http://docs.hp.com.

- **Is the disk the root disk, or part of the root volume group?**

  If the root disk is failing, the replacement process has extra steps to set up the boot area; in addition, you might have to boot from its mirror if the primary root disk has failed. If a failing root disk is not mirrored, you must reinstall to the replacement disk, or recover it from an Ignite-UX backup.

  To determine whether the disk is in the root volume group, use the `lvlnboot` command with the `–v` option. It lists the disks in the root volume group, and any special volumes configured on them. For example:

```
# lvlnboot –v
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
        /dev/disk/disk47_p2 -- Boot Disk
Boot: lvol1     on:     /dev/disk/disk47_p2
Root: lvol3     on:     /dev/disk/disk47_p2
Swap: lvol2     on:     /dev/disk/disk47_p2
Dump: lvol2     on:     /dev/disk/disk47_p2, 0
```

- **What logical volumes are on the disk, and are they mirrored?**

  After you replace the disk, you may have to restore data from backups. However, you will only have to recover data for a subset of the logical volumes in the volume group. Only the logical volumes that actually have physical extents on the disk are affected. In addition, if a logical volume is mirrored, there is likely a current copy of the data on the mirror, so it will not have to be recovered from backup.

  You can find the list of logical volumes using the disk with the `pvdisplay` command. With the `–v` option, `pvdisplay` shows a listing of all the physical extents on a physical volume and to what logical volume they belong. This list is rather long, so you should pipe it to `more` or send it to a file. For example:

```
# pvdisplay -v /dev/disk/disk3 | more
...
   --- Distribution of physical volume ---
   LV Name            LE of LV  PE for LV
   /dev/vg00/lvol5    50        50
   /dev/vg00/lvol6    245       245
 ...
```

From this example, you can see that logical volumes `/dev/vg00/lvol5` and `/dev/vg00/lvol6` have physical extents on this disk, so you only may have to restore `lvol5` and `lvol6`.

If `pvdisplay` fails, you can refer to any configuration documentation you created in advance, or use the `vgcfgdisplay` command, available from your HP support representative.

To determine if a logical volume is mirrored and if the mirror copies are current, use the `lvdisplay` command.

For each of the logical volumes affected, use `lvdisplay` to determine if the number of mirror copies is greater than zero. This verifies that the logical volume is mirrored. For example:

```
# lvdisplay /dev/vg00/lvol1
--- Logical volumes ---
LV Name                     /dev/vg00/lvol1
VG Name                     /dev/vg00
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               1
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            300
Current LE                  75
Allocated PE                150
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   off
Allocation                  strict/contiguous
IO Timeout (Seconds)        default
```

Since the number of mirror copies is not zero, the logical volume is mirrored.

Use `lvdisplay` again to determine which logical extents are mapped onto the suspect disk, and whether there is a current copy of that data *on another disk*. With the `-v` option, `lvdisplay` will show every logical extent, its mapping to any physical extents, and the status of those physical extents (stale or current).

This listing is likely to be quite long, so use `grep` confine the listing to the disk that is being replaced. For example:

```
# lvdisplay -v /dev/vg00/lvol1 | grep -e /dev/disk/disk3 -e '???'
   00000 /dev/disk/disk3  00000 current  /dev/disk/disk6  00000 current
   00001 /dev/disk/disk3  00001 current  /dev/disk/disk6  00001 current
   00002 /dev/disk/disk3  00002 current  /dev/disk/disk6  00002 current
   00003 /dev/disk/disk3  00003 current  /dev/disk/disk6  00003 current
   00004 /dev/disk/disk3  00004 current  /dev/disk/disk6  00004 current
   00005 /dev/disk/disk3  00005 current  /dev/disk/disk6  00005 current
...
```

In this example, all of `lvol1`'s physical extents on `/dev/disk/disk3` have a current copy elsewhere on the system, specifically on `/dev/disk/disk6`. If `/dev/disk/disk3` had been unavailable when the volume group was activated, its column would contain a '???' instead of the disk name.

Based on the gathered information, choose the appropriate procedure.

Replacing a Mirrored, Non-Boot Disk

Use this procedure if *all* the physical extents on the disk have copies on another disk, and your disk is not a boot disk. If the disk contains any unmirrored logical volumes or any mirrored logical volumes without an available and current mirror copy, use the procedure "Replacing an Unmirrored, Non-Boot Disk" (page 136).

For this example, the disk to be replaced is at lunpath hardware path 0/1/1/1.0x3.0x0, with device special files named `/dev/disk/disk14` and `/dev/rdisk/disk14`.

1. **Save the hardware paths to the disk.**

   Run the `ioscan` command and note the hardware paths of the failed disk.

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
======================================================================
disk 14 64000/0xfa00/0x0  esdisk  CLAIMED   DEVICE   offline HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
```

   In this example, the LUN instance number is 14, the LUN hardware path is 64000/0xfa00/0x0, and the lunpath hardware path is 0/1/1/1.0x3.0x0.

   When the failed disk is replaced, a new LUN instance and LUN hardware path will be created. To identify the disk once it is replaced, you must use the lunpath hardware path (in this case, 0/1/1/1.0x3.0x0).

2. **Halt LVM access to the disk.**

   If the disk is not hot-swappable, you must power down the system to replace it. By shutting down the system, you halt LVM access to the disk, so you can skip this step.

   Otherwise, detach the device using the `-a` option of the `pvchange` command:

```
# pvchange -a N /dev/disk/disk14
```

3. **Replace the disk.**

   For the hardware details on how to replace the disk, refer to the hardware administrator's guide for the system or disk array.

   If the disk is hot-swappable, simply replace it.

If the disk is not hot-swappable, shut down the system, turn off the power, and replace the disk. Reboot the system as normal.

4. **Notify the mass storage subsystem that the disk has been replaced.**

   If the system has not been rebooted to replace the failed disk, then you must run `scsimgr` before the new disk can be used as a replacement for the old disk. For example:

   ```
   # scsimgr replace_wwid -D /dev/rdisk/disk14
   ```

   This command allows the storage subsystem to replace the old disk's LUN World-Wide-Identifier (WWID) with the new disk's LUN World-Wide-Identifier. The storage subsystem will create a new LUN instance and new device special files for the replacement disk.

5. **Determine the new instance number for the disk.**

   Run `ioscan -m lun` to determine the new LUN instance created for the replacement disk. For example:

   ```
   # ioscan -m lun
   Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
   ======================================================================
   disk 14 64000/0xfa00/0x0  esdisk  NO_HW     DEVICE   offline HP MSA Vol
                     /dev/disk/disk14      /dev/rdisk/disk14
   ...
   disk 28 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
           0/1/1/1.0x3.0x0
                     /dev/disk/disk28      /dev/rdisk/disk28
   ```

   In this example, LUN instance 28 has been created for the new disk, with LUN hardware path 64000/0xfa00/0x1c, device special files `/dev/disk/disk28` and `/dev/rdisk/disk28`, at the same lunpath hardware path as the old disk, 0/1/1/1.0x3.0x0. Note that the old LUN instance 14 for the old disk now has no lunpath associated with it.

NOTE: If the system was rebooted to replace the failed disk, then the old disk will not be displayed by `ioscan -m lun`.

6. **Assign the old instance number to the replacement disk.**

   Run `io_redirect_dsf` to reassign the old LUN instance number to the new disk. For example:

   ```
   # io_redirect_dsf -d /dev/disk/disk14 -n /dev/disk/disk28
   ```

   This assigns the old LUN instance number (14) to the replacement disk. In addition, the device special files for the new disk are renamed to be consistent with the old LUN instance number. The following `ioscan -m lun` output shows the result:

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
======================================================================
disk 14 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
```

The LUN representation of the old disk with LUN hardware path 64000/0xfa00/0x0 has been removed. The LUN representation of the new disk with LUN hardware path 64000/0xfa00/0x1c has been reassigned from LUN instance 28 to LUN instance 14 and its device special files have been renamed as `/dev/disk/disk14` and `/dev/rdisk/disk14`.

7. **Put LVM configuration information on the disk.**

   Run `vgcfgrestore` to restore LVM configuration information to the added disk:

   ```
   # vgcfgrestore -n /dev/vgnn /dev/rdisk/disk14
   ```

8. **Restore LVM access to the disk.**

   In Step 2 above, if you did *not* reboot the system, reattach the disk by running the `pvchange` command with the `-a` option:

   ```
   # pvchange -a y /dev/disk/disk14
   ```

   If you did reboot the system, reattach the disk by running the `vgchange` command to reactivate the volume group and reattach any missing disks:

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

NOTE: The `vgchange` command with the `-a y` option can be run on a volume group that is deactivated or already activated. It attaches all paths for all disks in the volume group and resumes automatically recovering any disks in the volume group that had been offline or any disks in the volume group that have been replaced. Therefore, run `vgchange` only after all work has been completed on all disks and paths in the volume group, and it is desirable to attach them all.

Since all the data on the replaced disk was mirrored, you do not have to do anything else; LVM automatically synchronizes the data on the disk with the other mirror copies of the data.

Replacing an Unmirrored, Non-Boot Disk

Use this procedure if *any* of the physical extents on the disk do not have mirror copies elsewhere, and your disk is not a boot disk.

For this example, the disk to be replaced is at lunpath hardware path 0/1/1/1.0x3.0x0, with device special files named `/dev/disk/disk14` and `/dev/rdisk/disk14`.

1. **Save the hardware paths to the disk.**

   Run the `ioscan` command and note the hardware paths of the failed disk.

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
======================================================================
disk 14 64000/0xfa00/0x0  esdisk  CLAIMED   DEVICE   offline HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
```

   In this example, the LUN instance number is 14, the LUN hardware path is 64000/0xfa00/0x0, and the lunpath hardware path is 0/1/1/1.0x3.0x0.

   When the failed disk is replaced, a new LUN instance and LUN hardware path will be created. To identify the disk once it is replaced, you must use the lunpath hardware path (in this case, 0/1/1/1.0x3.0x0).

2. **Halt LVM access to the disk.**

   If the disk is not hot-swappable, you must power down the system to replace it. By shutting down the system, you halt LVM access to the disk, so you can skip this step.

   Otherwise, disable user and LVM access to all unmirrored logical volumes.

   First, disable *user* access to all unmirrored logical volumes. Halt any applications and unmount any file systems using these logical volumes. This prevents the applications or file systems from writing inconsistent data over the newly restored replacement disk.

   For *each* unmirrored logical volume using the disk:

   a. Use the `fuser` command to make sure no one is accessing the logical volume, either as a raw device or as a file system. If a user has files open in the file system or it is their current working directory, `fuser` will report their process IDs.
   b. If `fuser` reports some process IDs using the logical volume, use the `ps` command to map the list of process IDs to processes, and then determine whether you can halt those processes.
   c. If so, use `fuser` with the `-k` option to kill all processes accessing the logical volume.
   d. If the logical volume is being used as a file system, unmount it.

   For example, if the logical volume was `/dev/vg01/lvol1`:

```
# fuser -cu dev/vg01/lvol1
/dev/vg01/lvol1:    27815c(root)   27184c(root)
```

Look up processes 27815 and 27184:

```
# ps -fp27815 -p27184
     UID   PID  PPID  C    STIME TTY       TIME COMMAND
    root 27815 27184  0 09:04:05 pts/0     0:00 vi test.c
    root 27184 27182  0 08:26:24 pts/0     0:00 -sh
```

These processes are non-critical, so you can kill them:

```
# fuser -ku dev/vg01/lvol1
/dev/vg01/lvol1:    27815c(root)   27184c(root)
```

Finally, unmount the file system:

```
# umount /dev/vg01/lvol1
```

NOTE: If you cannot stop the applications using the logical volume, or you cannot unmount the file system, you must shut down the system.

After disabling user access to the unmirrored logical volumes, disable LVM access to the disk using the `-a` option of the `pvchange` command:

```
# pvchange -a N /dev/disk/disk14
```

3. **Replace the disk.**

   For the hardware details on how to replace the disk, refer to the hardware administrator's guide for the system or disk array.

   If the disk is hot-swappable, simply replace it.

   If the disk is not hot-swappable, shut down the system, turn off the power, and replace the disk. Reboot the system as normal.

4. **Notify the mass storage subsystem that the disk has been replaced.**

   If the system has not been rebooted to replace the failed disk, then you must run `scsimgr` before the new disk can be used as a replacement for the old disk. For example:

   ```
   # scsimgr replace_wwid -D /dev/rdisk/disk14
   ```

   This command allows the storage subsystem to replace the old disk's LUN World-Wide-Identifier (WWID) with the new disk's LUN World-Wide-Identifier. The storage subsystem will create a new LUN instance and new device special files for the replacement disk.

5. **Determine the new instance number for the disk.**

   Run `ioscan -m lun` to determine the new LUN instance created for the replacement disk. For example:

```
# ioscan -m lun
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
=============================================================================
disk 14 64000/0xfa00/0x0  esdisk  NO_HW     DEVICE   offline HP MSA Vol
                  /dev/disk/disk14      /dev/rdisk/disk14
...
disk 28 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk28      /dev/rdisk/disk28
```

   In this example, LUN instance 28 has been created for the new disk, with LUN hardware path 64000/0xfa00/0x1c, device special files `/dev/disk/disk28` and `/dev/rdisk/disk28`, at the same lunpath hardware path as the old disk, 0/1/1/1.0x3.0x0. Note that the old LUN instance 14 for the old disk now has no lunpath associated with it.

NOTE: If the system was rebooted to replace the failed disk, then the old disk will not be displayed by `ioscan -m lun`.

6. **Assign the old instance number to the replacement disk.**

   Run `io_redirect_dsf` to reassign the old LUN instance number to the new disk. For example:

```
# io_redirect_dsf -d /dev/disk/disk14 -n /dev/disk/disk28
```

   This assigns the old LUN instance number (14) to the replacement disk. In addition, the device special files for the new disk are renamed to be consistent with the old LUN instance number. The following `ioscan -m lun` output shows the result:

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
=============================================================================
disk 14 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
```

   The LUN representation of the old disk with LUN hardware path 64000/0xfa00/0x0 has been removed. The LUN representation of the new disk with LUN hardware path 64000/0xfa00/0x1c has been reassigned from LUN instance 28 to LUN instance 14 and its device special files have been renamed as `/dev/disk/disk14` and `/dev/rdisk/disk14`.

7. **Put LVM configuration information on the disk.**

   Run `vgcfgrestore` to restore LVM configuration information to the added disk:

```
# vgcfgrestore -n /dev/vgnn /dev/rdisk/disk14
```

8. **Restore LVM access to the disk.**

   In Step 2 above, if you did *not* reboot the system, reattach the disk by running the `pvchange` command with the `–a` option:

   ```
   # pvchange –a y /dev/disk/disk14
   ```

   If you did reboot the system, reattach the disk by running the `vgchange` command to reactivate the volume group and reattach any missing disks:

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

NOTE: The `vgchange` command with the `-a y` option can be run on a volume group that is deactivated or already activated. It attaches all paths for all disks in the volume group and resumes automatically recovering any disks in the volume group that had been offline or any disks in the volume group that have been replaced. Therefore, run `vgchange` only after all work has been completed on all disks and paths in the volume group, and it is desirable to attach them all.

9. **Recover any lost data.**

   LVM will recover all the mirrored logical volumes on the disk, and will start that recovery when the volume group is activated.

   For all the unmirrored logical volumes that you identified in Step 2, restore the data from backup, and reenable user access:

   - For raw volumes, restore the full raw volume using the utility that was used to create your backup. Then restart the application.
   - For file systems, you must re-create the file systems first. Use the `newfs` command:

     ```
     # newfs -F fstype /dev/vgnn/rlvolnn
     ```

     Use the logical volume's character device file for the `newfs` command. For file systems that had non-default configurations, consult the man page of `newfs` for the correct options.

     After creating the file system, mount it under the mount point that it previously occupied. When this is done, restore the data for that file system from your full backups.

TIP: To make the file system recreation step easier, record how they were originally created. You can change other file system parameters, such as those used to tune file system performance. The only critical feature is that the file system be at least as large as before the disk failure.

Replacing a Mirrored Boot Disk

There are two additional operations you must perform when replacing a mirrored boot disk:

1. You must initialize boot information on the replacement disk.
2. If the replacement requires rebooting the system, and the primary boot disk is being replaced, you must boot from the alternate boot disk.

Otherwise, the procedure is similar to that for replacing any other mirrored disk.

For this example, the disk to be replaced is at lunpath hardware path 0/1/1/1.0x3.0x0, with device special files named `/dev/disk/disk14` and `/dev/rdisk/disk14`. The system is an HP Integrity server, so the physical volume names must specify the HP-UX partition on the boot disk (`/dev/disk/disk14_p2` and `/dev/disk/disk14_p2`).

1. **Save the hardware paths to the disk.**

   Run the `ioscan` command and note the hardware paths of the failed disk.

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
======================================================================
disk 14 64000/0xfa00/0x0  esdisk  CLAIMED   DEVICE   offline HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
                  /dev/disk/disk14_p1   /dev/rdisk/disk14_p1
                  /dev/disk/disk14_p2   /dev/rdisk/disk14_p2
                  /dev/disk/disk14_p3   /dev/rdisk/disk14_p3
```

   In this example, the LUN instance number is 14, the LUN hardware path is 64000/0xfa00/0x0, and the lunpath hardware path is 0/1/1/1.0x3.0x0.

   When the failed disk is replaced, a new LUN instance and LUN hardware path will be created. To identify the disk once it is replaced, you must use the lunpath hardware path (in this case, 0/1/1/1.0x3.0x0).

2. **Halt LVM access to the disk.**

   If the disk is not hot-swappable, you must power down the system to replace it. By shutting down the system, you halt LVM access to the disk, so you can skip this step.

   Otherwise, detach the device using the `-a` option of the `pvchange` command:

```
# pvchange -a N /dev/disk/disk14_p2
```

NOTE: On an HP 9000 server, the boot disk is not partitioned, so the physical volume would refer to the entire disk, not the HP-UX partition. Use this command:

```
# pvchange -a N /dev/disk/disk14
```

3. **Replace the disk.**

   For the hardware details on how to replace the disk, refer to the hardware administrator's guide for the system or disk array.

   If the disk is hot-swappable, simply replace it.

   If the disk is not hot-swappable, shut down the system, turn off the power, and replace the disk. Reboot the system. You may run into two issues:

   - If you replaced the disk that you normally boot from, the replacement disk will not contain the information needed by the boot loader. In this case, interrupt the boot process, and boot from the mirror boot disk, which should be configured as the alternate boot path.
   - If there are only two disks in the root volume group, the system will probably fail its quorum check, as described in "Volume Group Activation Failures" (page 126). It may panic early in the boot process with the message:

     ```
     panic: LVM: Configuration failure
     ```

     In this situation, you must override quorum to boot successfully. Do this by interrupting the boot process and adding the option `-lq` to the boot command normally used by the system.

   For information on the boot process and how to select boot options, see *HP-UX System Administrator's Guide: Configuration Management*.

4. **Notify the mass storage subsystem that the disk has been replaced.**

   If the system has not been rebooted to replace the failed disk, then you must run `scsimgr` before the new disk can be used as a replacement for the old disk. For example:

   ```
   # scsimgr replace_wwid -D /dev/rdisk/disk14
   ```

   This command allows the storage subsystem to replace the old disk's LUN World-Wide-Identifier (WWID) with the new disk's LUN World-Wide-Identifier. The storage subsystem will create a new LUN instance and new device special files for the replacement disk.

5. **Determine the new instance number for the disk.**

   Run `ioscan -m lun` to determine the new LUN instance created for the replacement disk. For example:

```
# ioscan -m lun
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
=============================================================================
disk 14 64000/0xfa00/0x0  esdisk  NO_HW     DEVICE   offline HP MSA Vol
                  /dev/disk/disk14      /dev/rdisk/disk14
                  /dev/disk/disk14_p1   /dev/rdisk/disk14_p1
                  /dev/disk/disk14_p2   /dev/rdisk/disk14_p2
                  /dev/disk/disk14_p3   /dev/rdisk/disk14_p3
...
disk 28 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk28      /dev/rdisk/disk28
```

   In this example, LUN instance 28 has been created for the new disk, with LUN hardware path 64000/0xfa00/0x1c, device special files `/dev/disk/disk28` and `/dev/rdisk/disk28`, at the same lunpath hardware path as the old disk, 0/1/1/1.0x3.0x0. Note that the old LUN instance 14 for the old disk now has no lunpath associated with it.

NOTE: If the system was rebooted to replace the failed disk, then the old disk will not be displayed by `ioscan -m lun`.

6. **(HP Integrity servers only) Partition the replacement disk.**

   If your system is an HP Integrity server, partition the disk using the `idisk` command and a partition description file, and create the partition device files using `insf`, as described in "Mirroring the Boot Disk on HP Integrity Servers" (page 105).

7. **Assign the old instance number to the replacement disk.**

   Run `io_redirect_dsf` to reassign the old LUN instance number to the new disk. For example:

```
# io_redirect_dsf -d /dev/disk/disk14 -n /dev/disk/disk28
```

   This assigns the old LUN instance number (14) to the replacement disk. In addition, the device special files for the new disk are renamed to be consistent with the old LUN instance number. The following `ioscan -m lun` output shows the result:

```
# ioscan -m lun /dev/disk/disk14
Class I Lun H/W Path      Driver  S/W State H/W Type Health  Description
=============================================================================
disk 14 64000/0xfa00/0x1c esdisk  CLAIMED   DEVICE   online  HP MSA Vol
        0/1/1/1.0x3.0x0
                  /dev/disk/disk14      /dev/rdisk/disk14
                  /dev/disk/disk14_p1   /dev/rdisk/disk14_p1
                  /dev/disk/disk14_p2   /dev/rdisk/disk14_p2
                  /dev/disk/disk14_p3   /dev/rdisk/disk14_p3
```

The LUN representation of the old disk with LUN hardware path 64000/0xfa00/0x0 has been removed. The LUN representation of the new disk with LUN hardware path 64000/0xfa00/0x1c has been reassigned from LUN instance 28 to LUN instance 14 and its device special files have been renamed as well.

8. **Put LVM configuration information on the disk.**

   Run `vgcfgrestore` to restore LVM configuration information to the added disk:

   ```
   # vgcfgrestore -n /dev/vg00 /dev/rdisk/disk14_p2
   ```

NOTE: On an HP 9000 server, the boot disk is not partitioned, so the physical volume would refer to the entire disk, not the HP-UX partition. Use this command:

```
# vgcfgrestore -n /dev/vg00 /dev/rdisk/disk14
```

9. **Restore LVM access to the disk.**

   In Step 2 above, if you did *not* reboot the system, reattach the disk by running the `pvchange` command with the `-a` option:

   ```
   # pvchange -a y /dev/disk/disk14_p2
   ```

   On an HP 9000 server, use this command:

   ```
   # pvchange -a y /dev/disk/disk14
   ```

   If you did reboot the system, reattach the disk by running the `vgchange` command to reactivate the volume group and reattach any missing disks:

   ```
   # vgchange -a y /dev/vg00
   ```

NOTE: The `vgchange` command with the `-a y` option can be run on a volume group that is deactivated or already activated. It attaches all paths for all disks in the volume group and resumes automatically recovering any disks in the volume group that had been offline or any disks in the volume group that have been replaced. Therefore, run `vgchange` only after all work has been completed on all disks and paths in the volume group, and it is desirable to attach them all.

10. **Initialize boot information on the disk.**

    For an HP Integrity server, use the `mkboot` command to set up the boot area and update the autoboot file in the disk's EFI partition. These are described in step 5 and step 6 of "Mirroring the Boot Disk on HP Integrity Servers" (page 105).

For an HP 9000 server, use the `mkboot` command to set up the boot area and update the autoboot file, as described in step 4 and step 5 of "Mirroring the Boot Disk on HP 9000 Servers" (page 103).

Replacing an Unmirrored Boot Disk

With the failure of an unmirrored boot disk, you have lost the only copy of information that is required to boot the system. Unfortunately, you must reinstall to the replacement disk, or recover it from an Ignite-UX backup.

Warning and Error Messages

This section lists some of the warning and error messages reported by LVM. For each message, the cause is listed, and an administrator action is recommended.

TIP: Often an error message will contain the device number for a device, rather than the device file name. For example, you may see this message in `/var/adm/syslog/syslog.log`:

```
SCSI: Request Timeout -- lbolt: 329741615, dev: 1f022000
```

To map this error message to a specific disk, look under the `/dev` directory for a device file with a device number that matches the printed value. More specifically, search for a file whose minor number matches the lower six digits of the number following `dev:`. The device number in this example is 1f022000; its lower six digits are 022000, so search for that value using the following command:

```
# ll /dev/*dsk | grep 022000
brw-r-----   1 bin        sys         31 0x022000 Sep 22  2002 c2t2d0
crw-r-----   1 bin        sys        188 0x022000 Sep 25  2002 c2t2d0
```

All LVM Commands

Message Text:

```
vgcfgbackup: /etc/lvmtab is out of date with the running kernel:
Kernel indicates # disks for "/dev/vgname"; /etc/lvmtab has # disks.
Cannot proceed with backup.
```

Cause:

The number of current and active physical volumes, printed by `vgdisplay` as `Cur PV` and `Act PV`, are not the same. `Cur PV` and `Act PV` must always agree for the volume group. This error also indicates that the `/etc/lvmtab` file, which is used to

match physical volumes to a volume group, is out of date with the LVM data structures in memory and on disk.

Recommended Action:

Try to locate any missing disks. For each of the disk in the volume group, use `ioscan` and `diskinfo` to confirm that the disk is functioning properly.

lvchange(1M)

Message Text:

```
"m": Illegal option.
```

Cause:

The system does not have HP MirrorDisk/UX installed.

Recommended Action:

Install HP MirrorDisk/UX.

lvextend(1M)

Message Text:

```
lvextend: Not enough physical extents available.
Logical volume "/dev/vgname/lvname" could not be extended.
Failure possibly caused by strict allocation policy
```

Cause:

There is not enough space in the volume group to extend the logical volume to the requested size. This is typically caused by one of three situations:

1. There are not enough free physical extents in the volume group. Run `vgdisplay` to confirm the number of available physical extents, and multiply that number by the extent size to determine the free space in the volume group. For example:

```
# vgdisplay vg00
--- Volume groups ---
VG Name                     /dev/vg00
VG Write Access             read/write
VG Status                   available
Max LV                      255
Cur LV                      10
Open LV                     10
Max PV                      16
Cur PV                      1
Act PV                      1
Max PE per PV               4350
```

```
VGDA                          2
PE Size (Mbytes)              4
Total PE                      4340
Alloc PE                      3740
Free PE                       600
Total PVG                     0
Total Spare PVs               0
Total Spare PVs in use        0
```

In this example, the total free space is 600 * 4 MB, or 2400 MB.

2. The logical volume is mirrored with a strict allocation policy, and there are not enough extents on a separate disk to comply with the allocation policy. To confirm this, run `lvdisplay` to determine which disks the logical volume occupies, and then check whether there is sufficient space on the other disks in the volume group.
3. In a SAN environment, one of the disks was dynamically increased in size. LVM did not detect the asynchronous change in size.

Recommended Action:

1. Choose a smaller size for the logical volume, or add more disk space to the volume group.
2. Choose a smaller size for the logical volume, or add more disk space to the volume group. Alternatively, free up space on an available disk using `pvmove`.
3. Use the `vgmodify` command to detect the disk size change and incorporate the new space into the volume group.

Message Text:

```
"m": Illegal option.
```

Cause:

The system does not have HP MirrorDisk/UX installed.

Recommended Action:

Install HP MirrorDisk/UX.

lvlnboot(1M)

Message Text:

```
lvlnboot: Unable to configure swap logical volume.
Swap logical volume size beyond the IODC max address.
```

Cause:

The boot disk firmware cannot access the entire range of the swap logical volume. This happens with older host bus adapters when primary swap is configured past 4 GB on the disk.

Recommended Action:

Upgrade the system firmware or use a newer host bus adapter that supports block addressing. If neither of these actions is successful, reduce the size of the primary swap logical volume so that it does not exceed 4 GB.

pvchange(1M)

Message Text:

```
Unable to detach the path or physical volume via the pathname provided.
Either use pvchange(1M) -a N to detach the PV using an attached path
or detach each path to the PV individually using pvchange(1M) -a n
```

Cause:

The specified path is not part of any volume group, because the path has not been successfully attached to the otherwise active volume group it belongs to.

Recommended Action:

Check the specified path name to make sure it is correct. If the error occurred while detaching a physical volume, specify a different path that it was attached to before. If it is not clear whether any path was attached before, individually detach each path to the physical volume using `pvchange` with the *–a n* option.

Message Text:

```
Warning: Detaching a physical volume reduces the availability
of data within the logical volumes residing on that disk.
Prior to detaching a physical volume or the last available path to it,
verify that there are alternate copies of the data available on other
disks in the volume group. If necessary, use pvchange(1M) to reverse
this operation.
```

Cause:

This warning is advisory only and generated whenever a path or physical volume is detached.

Recommended Action:

None.

vgcfgbackup(1M)

Message Text:

```
Invalid LVMREC on Physical Volume.
```

Cause:

The LVM header on the disk is incorrect. This can happen when an existing LVM disk is overwritten with a command like `dd` or `pvcreate`. If the disk is shared between two systems, it is likely that one of the systems was not aware that the disk was already in a volume group. The corruption can also be caused by running `vgchgid` incorrectly when using BC split volumes.

Recommended Action:

Restore a known good configuration to the disk using `vgcfgrestore`. Be sure to use a valid copy dated before the first occurrence of the problem.

```
#  vgcfgrestore -n vgname pvname
```

vgcfgrestore(1M)

Message Text:

```
Cannot restore Physical Volume pvname
Detach the PV or deactivate the VG, before restoring the PV.
```

Cause:

The `vgcfgrestore` command was used to initialize a disk that already belongs to an active volume group.

Recommended Action:

Detach the physical volume or deactivate the volume group before attempting to restore the physical volume. If there is reason to believe that the data on the disk is corrupted, the disk can be detached and marked using `vgcfgrestore` then attached again without replacing the disk. This causes LVM to reinitialize the disk and synchronize any mirrored user data mapped there.

vgchange(1M)

Message Text:

```
Warning: couldn't query physical volume "pvname":
The specified path does not correspond to physical volume attached to this volume group
Warning: couldn't query all of the physical volumes.
```

Cause:

This error has the following possible causes:

1. The disk was missing when the volume group was activated, but was later restored. This typically occurs when a system is rebooted or the volume group is activated with a disk missing, uncabled, or powered down.
2. The disk LVM header was overwritten with the wrong volume group information. If the disk is shared between two systems, it is likely that one of the systems was not aware that the disk was already in a volume group. To confirm, check the volume group information using the `dump_lvmtab` command, available from your HP support representative, and look for inconsistencies. For example:

```
# dump_lvmtab -s | more
SYSTEM : 0x35c8cf58
TIME   : 0x3f9acc69 : Sat Oct 25 15:18:01 2003
FILE   : /etc/lvmtab
HEADER : version:0x03e8   vgnum:7
VG[00]  VGID:35c8cf58 3dd13164 (@0x00040c) pvnum:2 state:0 /dev/vg00
  (00)  VGID:35c8cf58 3dd13164 PVID:35c8cf58 3dd13164 /dev/dsk/c0t6d0
  (01)  VGID:35c8cf58 3dd13164 PVID:35c8cf58 3dda4694 /dev/dsk/c4t6d0
VG[01]  VGID:065f303f 3e63f01a (@0x001032) pvnum:92 state:0 /dev/vg01
  (00) !VGID:35c8cf58 3f8df316 PVID:065f303f 3e63effa /dev/dsk/c40t0d0
  (01) !VGID:35c8cf58 3f8df316 PVID:065f303f 3e63effe /dev/dsk/c40t0d4
  (02) !VGID:35c8cf58 3f8df316 PVID:065f303f 3e63f003 /dev/dsk/c40t1d0
...
```

   In this example, the volume group ids (VGID) for the disks in `/dev/vg01` are not consistent; inconsistencies are marked `!VGID`.

Recommended Action:

1. Use `ioscan` and `diskinfo` to confirm that the disk is functioning properly. Re-activate the volume group using the following command:

   ```
   # vgchange -a y vgname
   ```

2. There are several methods of recovery from this error. If you are not familiar with the commands outlined in the following procedures, contact your HP support representative for assistance.
   a. Restore a known good configuration to the disks using `vgcfgrestore`. Be sure to use a valid copy dated before the first occurrence of the problem.

      ```
      # vgcfgrestore -n vgname pvname
      ```

   b. Recreate the volume group and its logical volumes, restoring the data from the most current backup.
   c. Export and re-import the volume group, as described in "Exporting a Volume Group" (page 72) and "Importing a Volume Group" (page 73). For example:

```
# vgexport -m vgname.map -v -f vgname.file /dev/vgname
# mkdir /dev/vgname
# mknod /dev/vgname/group c 64 unique_minor_number
# vgimport -m vgname.map -v -f vgname.file /dev/vgname
```

Message Text:

```
vgchange: Couldn't set the unique id for volume group "/dev/vgname"
```

Cause:

There are multiple LVM group files with the same minor number.

Recommended Action:

List the LVM group files. If there are any duplicate minor numbers, export one of the affected volume groups, create a new group file with a unique minor number, and re-import the volume group. If you are not familiar with this procedure, contact your HP support representative for assistance.

```
# ll /dev/*/group
# vgexport -m vgname.map -v -f vgname.file /dev/vgname
# mkdir /dev/vgname
# mknod /dev/vgname/group c 64 unique_minor_number
# vgimport -m vgname.map -v -f vgname.file /dev/vgname
```

vgcreate(1M)

Message Text:

```
vgcreate: Volume group "/dev/vgname" could not be created:
VGRA for the disk is too big for the specified parameters.
Increase the extent size or decrease max_PVs/max_LVs and try again.
```

Cause:

The Volume Group Reserved Area at the front of each LVM disk cannot hold all the information about the disks in this volume group. This error typically occurs if you use disks larger than 100 GB.

Recommended Action:

Adjust the volume group creation parameters. Use the `-s` option of the `vgcreate` command to select an extent size larger than 4 MB, or use the `-p` option to select a smaller number of physical volumes. Refer to *vgcreate*(1M) for information on these options.

Message Text:

```
vgdisplay: Couldn't query volume group "/dev/vgname".
Possible error in the Volume Group minor number;
Please check and make sure the group minor number is unique.
vgdisplay: Cannot display volume group "/dev/vgname".
```

Cause:

This error has the following possible causes:

1. There are multiple LVM group files with the same minor number.
2. Serviceguard was previously installed on the system, and the `/dev/slvmvg` device file still exists.

Recommended Action:

1. List the LVM group files. If there are any duplicate minor numbers, export one of the affected volume groups, create a new group file with a unique minor number, and re-import the volume group. If you are not familiar with this procedure, contact your HP support representative for assistance.

   ```
   # ll /dev/*/group
   # vgexport -m vgname.map -v -f vgname.file /dev/vgname
   # mkdir /dev/vgname
   # mknod /dev/vgname/group c 64 unique_minor_number
   # vgimport -m vgname.map -v -f vgname.file /dev/vgname
   ```

2. Remove the `/dev/slvmvg` device file and recreate the `/etc/lvmtab` file using the following commands:

   ```
   # rm /dev/slvmvg
   # mv /etc/lvmtab /etc/lvmtab.old
   # vgscan -v
   ```

Message Text:

```
Warning: couldn't query physical volume "pvname":
The specified path does not correspond to physical volume
attached to this volume group
Warning: couldn't query all of the physical volumes.
```

Cause:

The possible causes of this error are described under the “vgchange(1M)” (page 149) error messages.

Recommended Action:

Refer to the recommended actions under the "vgchange(1M)" (page 149) error messages.

vgextend(1M)

Message Text:

```
vgextend: Not enough physical extents per physical volume.
Need: #, Have: #.
```

Cause:

The disk size exceeds the volume group maximum disk size. This limitation is defined when the volume group is created, as a product of the extent size specified with the `-s` option of `vgcreate` and the maximum number of physical extents per disk specified with the `-e` option. Typically, the disk is successfully added to the volume group, but not all of the disk is accessible.

Recommended Action:

The volume group extent size is not dynamic. Use the `vgmodify` command to adjust the maximum number of physical extents per disk. Alternatively, you can re-create the volume group with new values for the `-s` and `-e` options.

vgimport(1M)

Message Text:

```
Verification of unique LVM disk id on each disk in the volume group
/dev/vgname failed.
```

Cause:

There are two possible causes for this message:

1. The `vgimport` command used the `-s` option, and two or more disks on the system have the same LVM identifier; this can happen when disks are created with BC copy or cloned with `dd`.
2. LVM was unable to read the disk header; this can happen when you create new logical units on a SAN array.

Recommended Action:

1. Do not use the `-s` option to `vgimport`. Alternatively, use `vgchgid` to change the LVM identifiers on copied or cloned disks.
2. Retry the `vgimport` command.

/var/adm/syslog/syslog.log

Message Text:

```
LVM: VG 64 0xnn0000:
 Data in one or more logical volumes on PV nn 0x0nn000
 was lost when the disk was replaced.
This occurred because the disk contained the only copy of the data.
Prior to using these logical volumes, restore the data from backup.
```

Cause:

LVM cannot synchronize the data on a replaced disk automatically, as when LVM discovers an unmirrored logical volume residing on a disk that was just replaced. When all data on a disk is mirrored elsewhere and a copy is available, LVM automatically synchronizes the data on the replaced disk from the mirrors of the data on other disks.

Recommended Action:

Restore the contents of the logical volume from backup.

Message Text:

```
LVM: VG 64 0xnn0000: PVLink nn 0x0nn000 Detached.
```

Cause:

This message is advisory and generated whenever a disk path is detached.

Recommended Action:

None.

Message Text:

```
LVM: vg[nn] pv[nn] No valid MCR, resyncing all mirrored MWC LVs on the PV
```

Cause:

This message may be seen when importing a volume group from a previous release of HP-UX. The format of the mirror write cache (MWC) changed at HP-UX 11i Version 3, so if the volume group contains mirrored logical volumes using MWC, LVM converts the MWC at import time. It also performs a complete resynchronization of all mirrored logical volumes, which can take substantial time.

Recommended Action:

None. The message is advisory only.

A LVM Specifications and Limitations

This appendix discusses the product specifications.

NOTE: Do not infer that a system configured to these limits is usable.

Maximum data on a single HP-UX system is approximate 128 PB—This capacity is the product of the maximum number of volume groups on a system (256) and the maximum size of a single volume group (510 TB).

Maximum size of a single volume group is 510 TB—This capacity is the minimum of:

- 510 TB from the maximum number of disks (255) in a volume group times the maximum disk size (2 TB).
- 4080 TB from the maximum number of logical volumes in a volume group (255) times the maximum logical volume size (16 TB).

Maximum number of volume groups on a system is 256—A running HP-UX image can only communicate with 256 volume groups at one time. This limit comes from the 8-bit volume group index field in the high bits of the device minor number.

Maximum number of logical volumes in a volume group is 255—Each volume group can have a maximum of 255 logical volumes. This limit comes from the logical volume index in the lowest 8 bits of the minor number. One logical volume index is reserved for the control file for the group.

Maximum size of a logical volume is 16 TB—This limit comes from multiplying the maximum extent size (256 MB) by the maximum number of extents (64 K).

Maximum extent size is 256 MB—This is stored on-disk as a 32-bit signed integer.

Maximum physical volume size is 2 TB—A disk larger than 2 TB can be initialized for use with LVM, but only the first 2 TB are accessible.

Maximum number of physical volumes in a volume group is 255.

Maximum number of extents in a physical volume is 64 KB—This number is limited by a field in the disk header.

B Cheatsheet

This chapter contains a summary of the LVM commands and descriptions for their use.

Table B-1 LVM Command Summary

Command	Description
`extendfs`	Extends a file system: `# extendfs /dev/vg00/rlvol3`
`lvchange`	Changes the characteristics of a logical volume: `# lvchange -t 60 /dev/vg00/lvol3`
`lvcreate`	Creates a logical volume in a volume group: `# lvcreate -L 100 /dev/vg00`
`lvdisplay`	Displays information about logical volumes: `# lvdisplay -v /dev/vg00/lvol1`
`lvextend`	Adds a mirror to a logical volume: `# lvextend -m 1 /dev/vg00/lvol3`
`lvextend`	Increases the size of a logical volume: `# lvextend -L 120 /dev/vg00/lvol3`
`lvlnboot`	Prepares a logical volume to be a root, swap, or dump area: `# lvlnboot -d /dev/vg00/lvol2`
`lvmerge`	Merges split volumes into one logical volume: `# lvmerge /dev/vg00/lvol4b /dev/vg00/lvol4`
`lvreduce`	Decreases the size of a logical volume: `# lvreduce -L 100 /dev/vg00/lvol3`

Table B-1 LVM Command Summary *(continued)*

Command	Description
`lvreduce`	Decreases the number of mirror copies of a logical volume: `# lvreduce -m 0 /dev/vg00/lvol3`
`lvremove`	Removes logical volumes from a volume group: `# lvremove /dev/vg00/lvol6`
`lvrmboot`	Removes a logical volume link to root, swap, or dump: `# lvrmboot -d /dev/vg00/lvol2`
`lvsplit`	Splits a mirrored logical volume into two logical volumes: `# lvsplit /dev/vg00/lvol4`
`lvsync`	Synchronizes logical volume mirrors that are stale: `# lvsync /dev/vg00/lvol1`
`pvchange`	Changes the characteristics of a physical volume: `# pvchange -a n /dev/disk/disk2`
`pvck`	Performs a consistency check on a physical volume: `# pvck /dev/disk/disk47_p2`
`pvcreate`	Creates a physical volume be to used as part of a volume group: `# pvcreate /dev/rdisk/disk2`
`pvdisplay`	Displays information about a physical volume: `# pvdisplay -v /dev/disk/disk2`
`pvmove`	Moves extents from one physical volume to another: `# pvmove /dev/disk/disk2 /dev/disk/disk3`

Table B-1 LVM Command Summary *(continued)*

Command	Description
pvremove	Removes LVM data structures from a physical volume: **# pvremove /dev/rdisk/disk2**
vgcfgbackup	Saves LVM configuration for a volume group: **# vgcfgbackup vg00**
vgcfgrestore	Restores the LVM configuration: **# vgcfgrestore -n /dev/vg00 /dev/rdisk/disk2**
vgchange	Turns a volume group off or on: **# vgchange -a y /dev/vg00**
vgchgid	Changes the volume group ID of a physical volume: **# vgchgid /dev/rdisk/disk3**
vgcreate	Creates a volume group: **# vgcreate /dev/vg01 /dev/disk/disk2 /dev/disk/disk3**
vgdisplay	Displays information about a volume group: **# vgdisplay -v /dev/vg00**
vgextend	Extends a volume group by adding a physical volume: **# vgextend /dev/vg00 /dev/disk/disk2**
vgexport	Removes a volume group from the system: **# vgexport /dev/vg01**

Table B-1 LVM Command Summary *(continued)*

Command	Description
vgimport	Adds an existing volume group to the system: **# mkdir /dev/vg04** **# mknod /dev/vg04/group c 640x0*n*0000** **# vgimport -v /dev/vg04** (*n* is a unique number across all volume groups.)
vgmodify	Modifies configuration parameters of a volume group: **# vgmodify -v -t -n -r /dev/disk/disk3**
vgscan	Scans the system disks for volume groups: **# vgscan -v**
vgreduce	Reduces a volume group by removing one or more physical volumes from it: **# vgreduce /dev/vg00 /dev/disk/disk2**
vgremove	Removes the definition of a volume group from the system and the disks: **# vgremove /dev/vg00 /dev/disk/disk2**
vgsync	Synchronizes all mirrored logical volumes in the volume group: **# vgsync vg00**

C Glossary

This appendix contains a glossary of terms useful for setting up an LVM configuration.

Allocation Policy — The LVM allocation policy governing how disk space is distributed to logical volumes and how extents are laid out on an LVM disk. LVM allocates disk space in terms of strict vs. non-strict and contiguous vs. noncontiguous. Strict allocation requires that mirror copies reside on different LVM disks. Contiguous allocation requires that no gaps exist between physical extents on a single disk.

Disk Spanning — The allocation of a logical volume across multiple disks, allowing the volume size to exceed the size of a single disk.

I/O Channel Separation — A configuration of disks useful for segregating highly I/O-intensive areas. For example, you might have a database on one channel and file systems on another. When mirroring logical volumes using HP MirrorDisk/UX, you can spread the mirrored copies over different I/O channels to increase system and data availability.

Logical Volume — A virtual storage device of flexible size that can hold a file system, raw data, dump area, or swap. Because its data are distributed logically (rather than physically), a single logical volume can be mapped to one LVM disk or span multiple disks. A logical volume appears to the administrator as though it was a single disk.

Logical Extents — Fixed-size addressable areas of space on a logical volume. The basic allocation unit for a logical volume, a logical extent is mapped to a physical extent; thus, if the physical extent size is 4 MB, the logical extent size will also be 4 MB. The size of a logical volume is determined by the number of logical extents configured.

Logical Volume Manager — An operating system software module that implements virtual (logical) disks to extend, mirror, and improve the performance of physical disk access.

Mirroring — Simultaneous replication of data (up to three copies), ensuring a greater degree of data availability. LVM can map identical logical volumes to multiple LVM disks, thus providing the means to recover easily from the loss of one copy (or two copies in the case of double mirroring) of data. Mirroring can provide faster access to data for applications using more data reads than writes. Mirroring requires the MirrorDisk/UX product.

Physical Extents — Fixed-size addressable areas of space on an LVM disk. The basic allocation unit for a physical volume, physical extents are by default 4 MB. Physical extents map to areas on logical volumes called logical extents.

Physical Volume — A disk that has been initialized by LVM for inclusion in a volume group; also called an LVM disk. As with standard disks, an LVM disk (physical volume) is accessed via a raw device file (for example, `/dev/rdisk/disk3`). Use the HP SMH or the `pvcreate` command to initialize a disk as a physical volume.

Physical Volume Group — A subset of physical volumes within a volume group, each with a separate I/O channel or interface adapter to achieve higher availability of mirrored data.

Quorum — The requirement that a certain number of LVM disks be present in order to change or activate a volume group. To activate a volume group, quorum requires the number of available LVM disks to be *more than* half the number of configured LVM disks that were present when the volume group was last active. To make a configuration change, the quorum requirement is *at least* half. If there is no quorum, LVM prevents the operation.

Quorum is checked both during configuration changes (for example, when creating a logical volume) and at state changes (for example, if a disk fails). Quorum ensures the consistency and integrity of the volume groups. The `vgchange` command with the `-q n` option can be used to override quorum check, but this should be used with caution.

Synchronization The process of updating stale (non-current) copies of mirrored logical extents by copying data from a fresh (current) copy of the logical volume. Synchronization keeps mirrored logical volumes consistent by ensuring that all copies contain the same data.

Volume Group A collection of one or more LVM disks from which disk space may be allocated to individual logical volumes. A disk can belong to only one volume group. A volume group is accessed through the group file (for example, `/dev/vg01/group`) in that volume group's directory. Use HP SMH or the `vgcreate` command to create a volume group.

Index

M

N

P

Q

R

S