VERITAS Volume Manager (VxVM)
System Administrator's Guide

0890471-000
April 15, 1995

.

Revision History:	Level:	Effective With:
Original Release -- April 15, 1995	000	Veritas[R] Release 1.3

Related Documents:

Pubs No.	Title
0890472	VERITAS[R] Volume Manager (VxVM) User's Guide

Printed in U. S. A.

VERITAS® Volume Manager (VxVM®)

System Administrator's Guide

Release 1.3

SVR4.2MP
March, 1994
P/N 10-103003-1300

Copyrights

VERITAS Volume Manager Release 1.3

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Important Note to Users

While every effort has been made to ensure the accuracy of all information in this document, VERITAS assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident, or any other cause. VERITAS further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. VERITAS disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, *including implied warranties of merchantability or fitness for a particular purpose.*

VERITAS makes no representation that the interconnection of products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting or license to make, use or sell equipment constructed in accordance with this description.

VERITAS reserves the right to make changes without further notice to any products herein to improve reliability, function, or design.

Trademarks

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Co. Ltd.

VERITAS and the VERITAS logo are trademarks or registered trademarks of VERITAS Software Corporation in the USA and other countries.

Other trademarks mentioned in this document are trademarks or registered trademarks of their respective holders.

Contents

1-1

2-1

A-1

B-1

Preface

Audience

The *VERITAS Volume Manager® System Administrator's Guide* is for system administrators responsible for installing, configuring, and maintaining systems under the control of the VERITAS Volume Manager *(VxVM®)*. This guide assumes that the user has a:

- working knowledge of the UNIX® operating system
- basic understanding of system administration
- basic understanding of volume management

Scope

The purpose of this guide is to provide the user with a thorough knowledge of the procedures and concepts involved with volume management and system administration using the Volume Manager.

Organization

The *VERITAS Volume Manager System Administrator's Guide* is comprised of five chapters and three appendixes, organized as follows:

- Chapter 1—Introduction to VxVM

Chapter 1 contains an overview of VxVM, including information on how to use the VxVM to more effectively manage system resources and optimize system performance.

- Chapter 2—VxVM Performance Management

Chapter 2 suggests performance management and configuration guidelines for use with VxVM.

- Chapter 3—Disk Administration

Chapter 3 discusses disk and disk group administration using the Volume Manager. Disk administration tools are discussed, including the `vxdiskadm` menus, and `vxdisk`.

- Chapter 4—Volume Administration

Chapter 4 discusses the VxVM volume utilities, subdisks and plexes.

- Appendix A—Error Messages

Appendix A contains a list of error messages generated by VxVM, a description of what the messages mean, and user actions the system administrator should take should these messages occur.

- Appendix B—Recovery

Appendix B describes the procedures used to preserve and recover data. This appendix discusses ways to prevent data loss due to disk failure and to prevent loss of system availability due to failure of a key disk (a disk involved with system operation).

Related Documents

The following documents provide information related to the Volume Manager:

- *VERITAS Volume Manager (VxVM) User's Guide* provides introductory and conceptual information about the VERITAS Volume Manager (VxVM), and a user's guide to the VERITAS Visual Administrator graphical user interface. The command line and menu (vxdiskadm) interfaces to the Volume Manager are also documented.
- The *VERITAS File System (VxFS) System Administrator's Guide* provides detailed information on the VxFS file system.

Using This Guide

This guide includes instructions for performing Volume Manager system administrator functions. An explanation of relevant terms and concepts is also included.

Volume Manager administration functions can be performed via a set of fairly common commands, a single automated command, menus, or various graphical objects (such as icons). This guide focuses on the approaches to Volume Manager administration offered by:

- the Volume Manager Support Operations (vxdiskadm) menus
- the `vxassist` commands
- the VxVM command set

The VxVM User Interface

The user interface for the Volume Manager falls into various categories, including a simple command set (`vxassist`), a command line interface, a menu-driven interface (`vxdiskadm`), and the Visual Administrator, a graphical user interface. The user decides which of the available interfaces to use.

The Command Line Interface

Examples of the command line interface are presented in this guide in two forms: generic and example-specific.

The syntax for the generic interactive removal of the file *samplefile* is:

```
rm -i file_name
```

where *file_name* is the name of the appropriate file.

The syntax for the example-specific interactive removal of the file `samplefile` is:

```
rm -i samplefile
```

where the name of the file is “samplefile.”

Both of the preceding examples illustrate the same syntax, which is the removal of a file; specifying the `-i` option prompts the user for confirmation of the removal of a write-protected file. Most of the Volume Manager commands must be run by a user with appropriate privileges. For more information about user privileges, refer to the *VERITAS Volume Manager User's Guide.*

Detailed descriptions of the VxVM utilities, the options for each, and details on how to use them are located in the reference manual pages.

The Menu-Driven Interfaces

The `vxdiskadm` menu allows the administrator to select a disk administration operation from a main menu. `vxdiskadm` then provides detailed screens to lead the administrator through the selected disk administration procedure.

Conventions

The following table describes the typographic conventions used in this guide.

Table P-1 Typographic Conventions

Typeface or Symbol	**Meaning**	**Example**
`courier`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`
`courier bold`	User input, contrasted with on-screen computer output	`system%` **`su`** `Password:`
`courier italics`	Command-line placeholders or variables to be substituted with a real name or value	To delete a file, type `rm` *`filename`*.
italics	Manual titles, new terms, or words to be emphasized	Refer to Chapter 7 in *the User's Guide*.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
Code samples are included in boxes and may display the following:		
`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	
#	Superuser prompt, all shells	

Description of the Volume Manager 1

1.1 Introduction

To use the Volume Manager (VxVM) effectively, you must have a fairly good understanding of its principles of operation. This chapter provides information needed to understand the Volume Manager.

1.2 What the Volume Manager Does

The Volume Manager builds virtual devices called *volumes* on top of physical disks. Volumes are accessed by a UNIX file system, a database, or other applications in the same way physical disk partitions would be accessed. Volumes are composed of other virtual objects that can be manipulated to change the volume's configuration. Volumes and their virtual components are referred to as *Volume Manager objects*. Volume Manager objects can be manipulated in a variety of ways to optimize performance, provide redundancy of data, and perform backups or other administrative tasks on one or more physical disks without interrupting system users. As a result, data availability and disk subsystem throughput are improved.

To understand the Volume Manager, you must first understand the relationships between physical objects and Volume Manager objects.

1.2.1 Physical Objects

To perform disk management tasks using the Volume Manager, you must understand two physical objects:

- Physical disks
- Partitions

1.2.1.1 Physical Disks

A *physical disk* is the underlying storage device (media), which may or may not be under Volume Manager control. A physical disk can be accessed using a device name such as `c#t#d#`, where `c#` is the controller, `t#` is the target ID, and `d#` is the disk number. The disk in Figure 1-1 is disk number 0 with a target ID of 0, and it is connected to controller number 0 in the system.

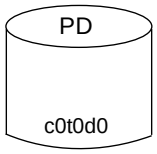

Figure 1-1 Example of a Physical Disk

1.2.1.2 Partitions

A physical disk can be divided into one or more *partitions*. The partition number, or `s#`, is given at the end of the device name. Note that a partition could take up an entire physical disk, such as the partition shown in Figure 1-2.

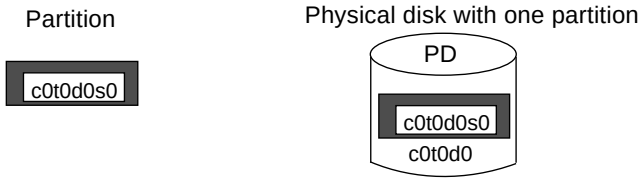

Figure 1-2 Example of a Partition

The relationship between physical objects and Volume Manager objects is established when you place a partition from a physical disk under Volume Manager control.

1.2.2 Volume Manager Objects

There are several Volume Manager objects that must be understood before you can use the Volume Manager to perform disk management tasks:

- VM disks
- Disk groups
- Subdisks
- Plexes
- Volumes

1.2.2.1 VM Disks

A *VM disk* is a contiguous area of disk space from which the Volume Manager allocates storage. When you place a partition from a physical disk under Volume Manager control, a VM disk is assigned to the partition. A VM disk has a one-to-one relationship with a partition. A VM disk is accessed using a *disk media name*, which you can supply (or else the Volume Manager assigns one). Figure 1-3 shows a VM disk with a disk media name of `disk01` that is assigned to the partition `c0t0d0s0`.

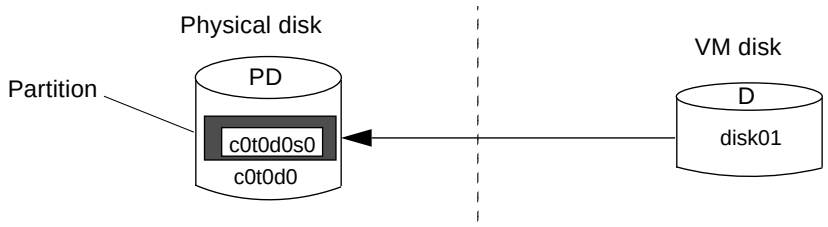

Figure 1-3 Example of a VM Disk

With the Volume Manager, applications access volumes (created on VM disks) rather than partitions. The Volume Manager therefore initializes each new disk with the fewest number of partitions possible (generally 2 partitions per physical disk).

1.2.2.2 Disk Groups

A *disk group* is a collection of VM disks that share a common configuration. A disk group is a set of records containing detailed information about existing VxVM objects, their attributes, and their relationships. The default disk group is `rootdg` (the root disk group). Additional disk groups can be created, as necessary. A given volume must be configured from disks belonging to the same disk group. Disk groups allow the administrator to group disks into logical collections for administrative convenience. A disk group and its components can be moved as a unit from one host machine to another.

1.2.2.3 Subdisks

A *subdisk* is a set of contiguous disk blocks; subdisks are the basic units in which the Volume Manager allocates disk space. Each subdisk represents a specific portion of a physical disk. A VM disk can be divided into one or more subdisks. Since the default name for a VM disk is `disk##` (such as `disk01`), the default name for a subdisk is `disk##-##`. So, for example, `disk01-01` would be the name of the first subdisk on the VM disk named `disk01`.

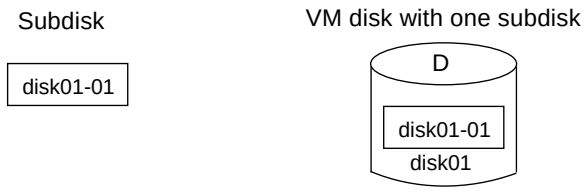

Figure 1-4 Example of a Subdisk

If a VM disk that is assigned to a partition contains more than one subdisk, then that partition is split into the number of subdisks contained in the VM disk. The example given in Figure 1-5 shows a VM disk, with three subdisks, that is assigned to one partition.

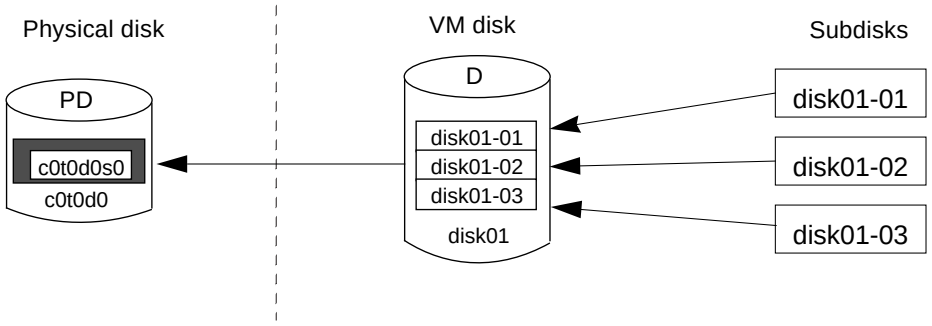

Figure 1-5 Example of Three Subdisks Assigned to One Partition

Any VM disk space that is not part of a subdisk is considered to be free space, which can be used to create new Volume Manager objects.

1.2.2.4 Plexes

The Volume Manager uses subdisks to build virtual devices called *plexes* (also referred to as *mirrors*). A plex consists of one or more subdisks that represent portions of one or more physical disks. There are two ways that data can be organized on the subdisks that constitute a plex:

- concatenation
- striping

Concatenation

Concatenation maps data in a linear manner onto one or more subdisks in a plex. If you were to access all the data in a concatenated plex sequentially, you would first access the data in the first subdisk from beginning to end, then access the data in the second subdisk from beginning to end, and so forth until the end of the last subdisk.

The subdisks in a concatenated plex do not necessarily have to be physically contiguous and can belong to more than one VM disk. Concatenation using subdisks that reside on more than one VM disk is also called *spanning*.

Figure 1-6 illustrates concatenation with one subdisk.

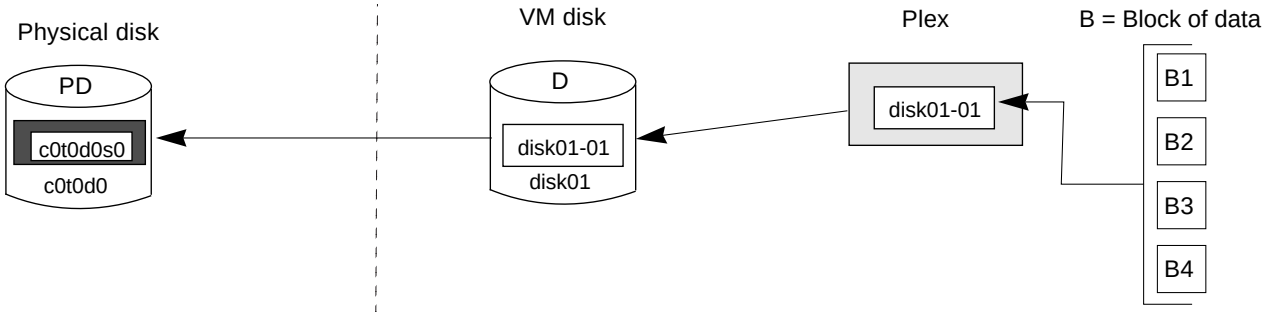

Figure 1-6 Example of Concatenation

Concatenation with more than one subdisk is useful when there is insufficient contiguous space for the plex on any one disk. Such concatenation is also useful for load balancing between disks, and for head movement optimization on a particular disk.

Figure 1-7 shows how data would be spread over two subdisks in a spanned plex.

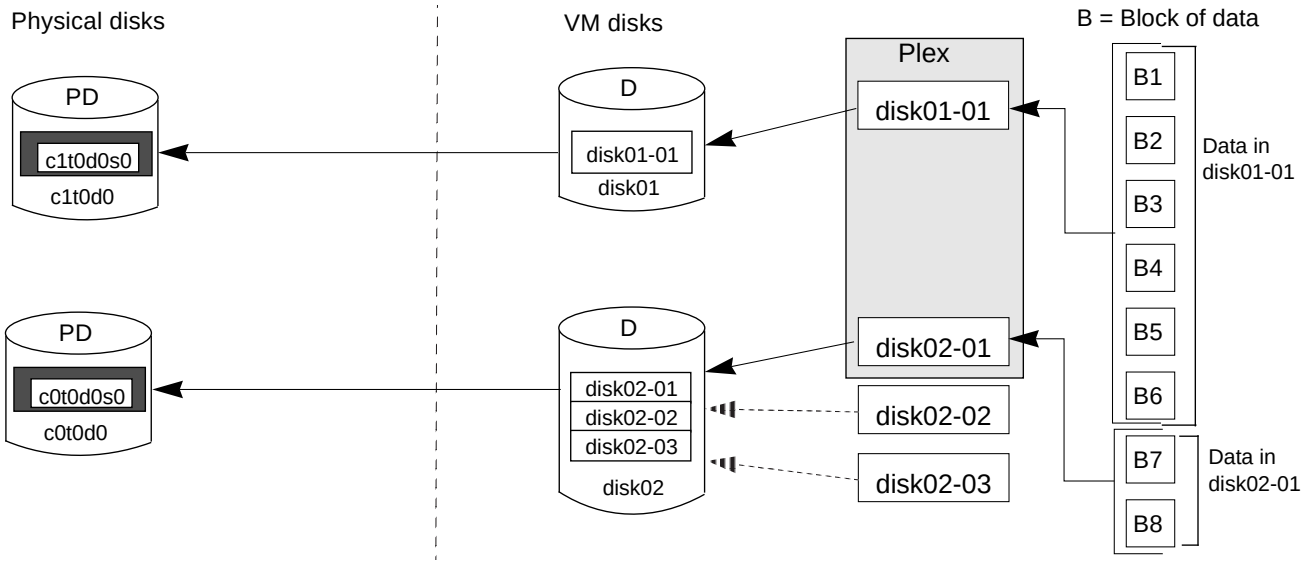

Figure 1-7 Example of Spanning

Since the first six blocks of data (`B1` through `B6`) took up most or all of the room on the partition that VM `disk01` is assigned to, subdisk `disk01-01` is alone on VM disk `disk01`. However, the last two blocks of data, `B7` and `B8`, take up only a portion of the room on the partition that VM `disk02` is assigned to. That means that the remaining free space on VM `disk02` can be put to other uses. In this example, the Volume Manager has created subdisks `disk02-02` and `disk02-03` for some other disk management task (such as mirroring or striping).

Striping

Striping maps data so that the data is interleaved among 2 or more physical disks. More specifically, a striped plex contains 2 or more subdisks, spread out over 2 or more physical disks. The subdisks are grouped into "columns," with one column per physical disk. Each column contains one or more subdisks.

Data is allocated in equal sized blocks ("stripe blocks") that are interleaved between the columns. For example, if there are 3 columns in the striped plex and 6 stripe blocks, the first and fourth stripe block would be allocated in column 1; the second and fifth stripe block would be allocated in column 2; and the third and sixth stripe block would be allocated in column 3. Viewed in sequence, if you were striping data over three physical disks, the first stripe block would go in the first column, the second block would go in the second column, and the third block would go in the third column. The fourth stripe block would then go back to the first column, the fifth block would go in the second column, and the sixth block would go in the third column.

Caution – Striping a volume, or splitting a volume across multiple disks, increases the chance that a disk failure will result in failure of that volume. For example, if five volumes are striped across the same five disks, then failure of any one of the five disks will require that all five volumes be restored from a backup. If each volume were on a separate disk, only one volume would have to be restored. Use mirroring to substantially reduce the chances that a single disk failure will result in failure of a large number of volumes.

Striping is useful if you need large amounts of data to be written to or read from the physical disks quickly by using parallel data transfer to multiple disks. Striping is also a way to balance the I/O load from multi-user applications across multiple disks.

Figure 1-8 shows a simple striped plex with 3 equal sized, single-subdisk columns. There is one column per physical disk.

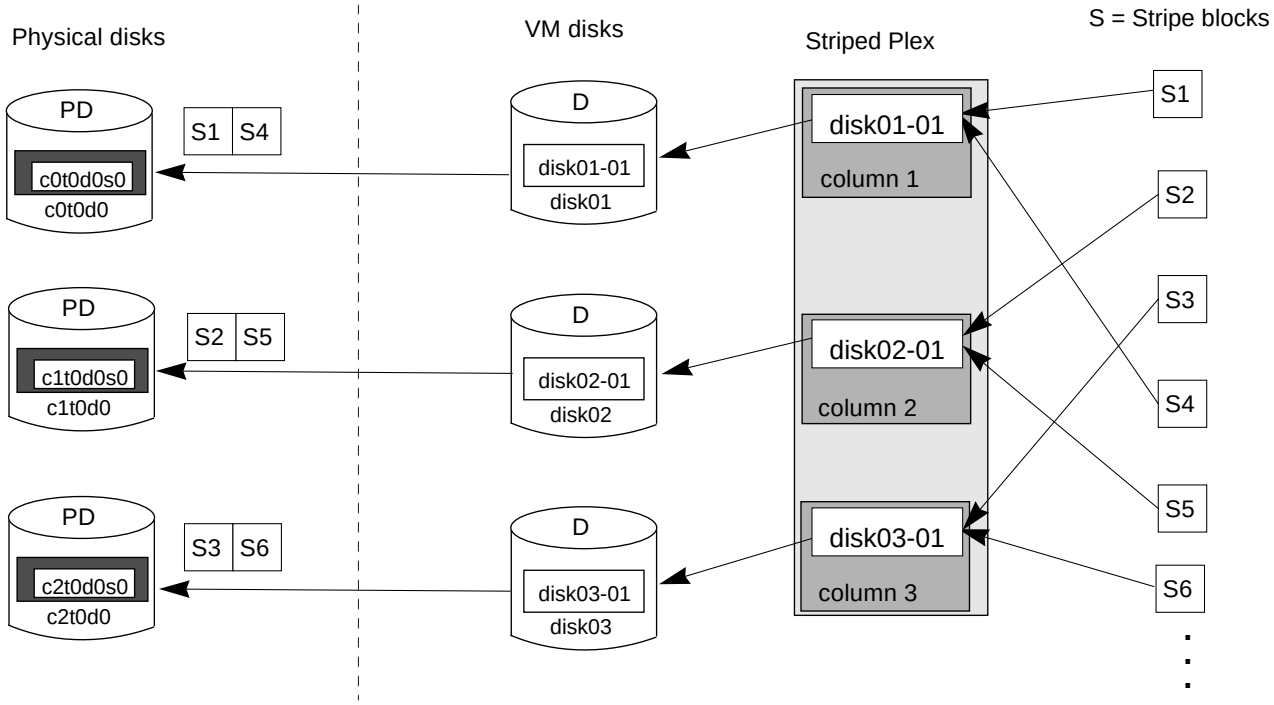

Figure 1-8 Example of Striped Plex with One Subdisk per Column

Although the example in Figure 1-8 shows three subdisks that take up all of the VM disks, it is also possible for each subdisk in a striped plex to take up only a portion of the VM disk, thereby leaving free space for other disk management tasks.

Figure 1-9 shows a striped plex with 3 columns containing subdisks of different sizes. Each of the columns contains a different number of subdisks. There is one column per physical disk. Although striped plexes are usually created using a single subdisk from each of the VM disks being striped across, it is also possible to allocate space from different regions of the same disk or from another disk (if the plex is grown, for instance).

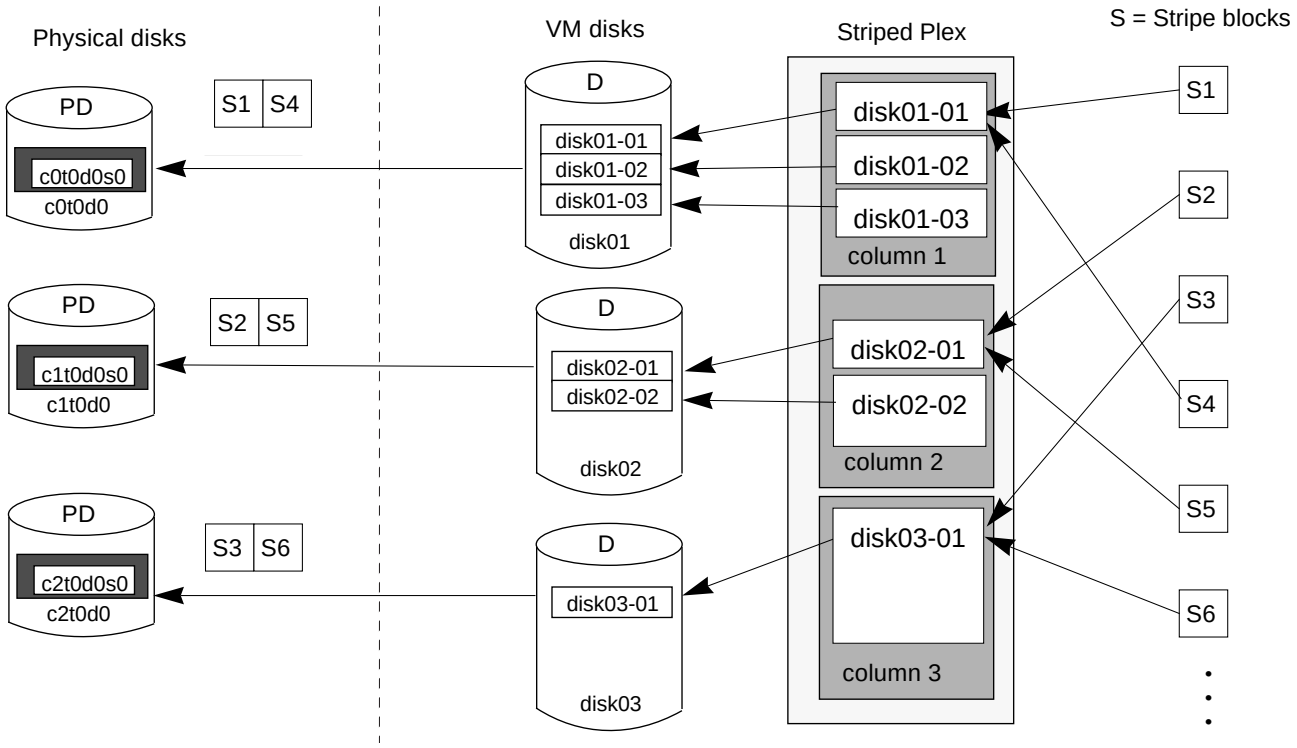

Figure 1-9 Example of Striped Plex with Multiple Subdisks per Column

1.2.2.5 Volumes

A *volume* is a virtual disk that appears to applications, data bases and file systems like a physical disk partition, but a volume does not have the physical limitations of a physical disk partition. A volume consists of one or more plexes, each holding a copy of the data in the volume. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area thereof. The configuration of a volume can be changed, using the Volume Manager interfaces, without causing disruption to applications or file systems that are using the volume. A volume can be mirrored, spanned across disks, moved to use different storage, and striped.

A volume can contain from one to eight plexes. The Volume Manager uses `vol##` as the default naming convention for volumes, and `vol##-##` as the default naming convention for plexes in a volume. A volume with one plex contains one copy of the data and would look like the volume shown in Figure 1-10. Note that all subdisks within a volume must belong to the same disk group.

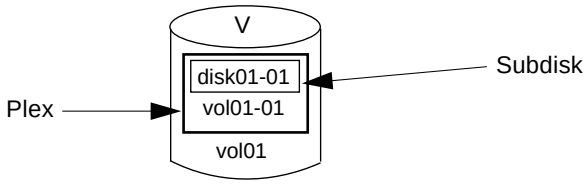

Figure 1-10 Example of Volume with One Plex

Note that volume `vol01` in Figure 1-10 has the following characteristics:

- it contains one plex named `vol01-01`
- the plex contains one subdisk named `disk01-01`
- the subdisk `disk01-01` is allocated from VM disk `disk01`

A volume with two or more plexes contains mirror images of the data and would look like the volume shown in Figure 1-11.

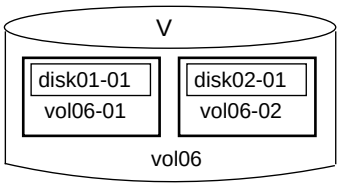

Figure 1-11 Example of Volume with Two Plexes

Note that volume `vol06` in Figure 1-11 has the following characteristics:

- it contains two plexes named `vol06-01` and `vol06-02`
- each plex contains one subdisk
- each subdisk is allocated from a different VM disk (`disk01` and `disk02`)

Figure 1-12 shows how a volume would look if it were set up for the simple striped configuration given in Figure 1-8.

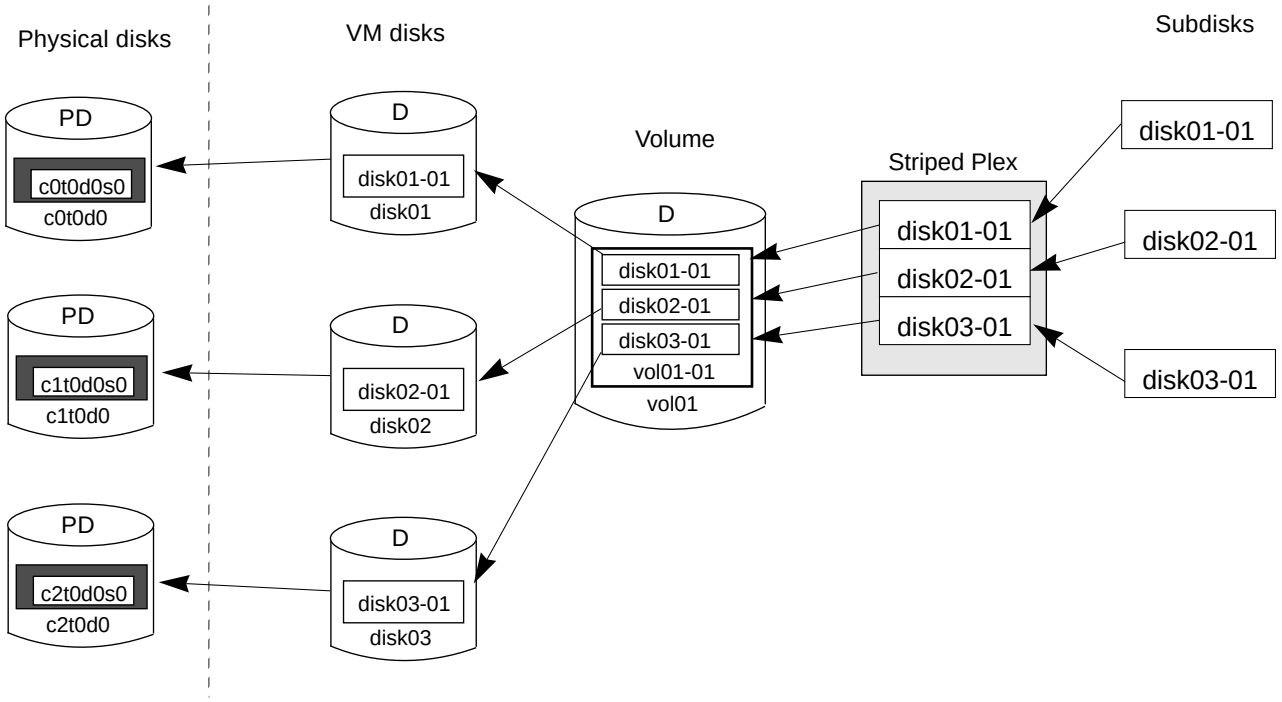

Figure 1-12 Example of Volume in a Striped Configuration

Mirroring

Mirroring is a technique of using multiple mirrors (plexes) to duplicate the information contained in a volume. In the event of a physical disk failure, the mirror on the failed disk becomes unavailable, but the system continues to operate using the unaffected mirrors. Although a volume can have a single plex, at least two plexes are required to provide redundancy of data. Each of these plexes should contain disk space from different disks in order for the redundancy to be effective.

When striping or spanning across a large number of disks, failure of any one of those disks will generally make the entire plex unusable. The chance of one out of several disks failing is sufficient to make it worthwhile to consider mirroring in order to improve the reliability of a striped or spanned volume.

1.2.3 Dirty Region Logging

Dirty Region Logging (DRL) is an optional property of a volume, used to provide a speedy recovery of mirrored volumes after a system failure.

DRL keeps track of the regions that have changed due to I/O writes to a mirrored volume. If DRL is not used and a system failure occurs, all mirrors of the volumes must be restored to a consistent state by copying the full contents of the volume between its mirrors. This process can be lengthy and I/O intensive. It may also be necessary to recover the areas of volumes that are already consistent.

DRL logically divides a volume into a set of consecutive regions. It keeps track of volume regions which are being written to. A dirty region log is maintained which contains a status bit for each region of the volume. *Log subdisks* are subdisks that are defined for and added to a mirror that is to become part of a volume that has DRL enabled. The log subdisk size must be chosen from an even number between 2 and 10 sectors.

For any write operation to the volume, before writing the data, the regions being written are marked dirty in the log. If a write causes a region to become dirty when it was previously clean, the log is written to disk before the write operation can occur.

On system restart, the Volume Manager will recover only those regions of the volume which are marked as dirty in the dirty region log.

The dirty bit for a region is not cleared immediately after writing the data to the region. Instead it is left untouched until the corresponding volume region becomes the least recently used.

Note – DRL adds a small I/O overhead for most access patterns.

1.3 VxVM Rootability

The Volume Manager provides the capability of placing the `root` and initial `swap` devices under VxVM control. Once under VxVM control, the `root` and `swap` devices appear as volumes and provide the same characteristics as other VxVM volumes. A volume that is configured for use as a swap area is referred to as a *swap volume*, while a volume that contains the root file system is referred to as a *root volume*.

The `rootvol` and `swapvol` volumes, as well as other parts of the `root` disk (for example, `/usr`) required for a successful boot of the system, can be mirrored. This provides complete redundancy and recovery capability in the event of disk failure. Without the Volume Manager rootability, the loss of the `root`, `swap` or `usr` partition would result in the system being unable to be booted from surviving disks.

By mirroring drives critical to booting, you ensure that no single disk failure will leave your system unusable. Therefore, a suggested configuration would be to use the `vxdiskadm` command menu to mirror the critical disk onto another available disk. In case of a failure of the disk containing the `root` partition, reboot the system from the disk containing the root mirror. For more information on mirroring the boot disk and system recovery procedures, see the “Recovery” appendix.

1.3.1 Booting with Root Volumes

Ordinarily, when the operating system is booted, the `root` file system and `swap` area need to be available for use very early in the boot procedure, which is long before user processes can be run to load the Volume Manager configuration and start volumes. The `root` and `swap` device configurations must be completed prior to starting the Volume Manager. Starting `vxconfigd` as part of the `init` process, is too late to configure volumes for use as a `root` or `swap` device.

To circumvent this restriction, the mirrors of the `rootvol` and `swapvol` volumes can be accessed by the system during startup. During startup, the system sees the `rootvol` and `swapvol` volumes as regular partitions and accesses them using standard partition numbering. Therefore, `rootvol` and the `swapvol` volumes must be created from contiguous disk space that is also mapped by a single partition for each. Due to this restriction, it is not possible to stripe or span `rootvol` and `swapvol` volumes.

1.3.2 Boot-time Volume Restrictions

The `rootvol` and `swapvol` volumes differ from other volumes in that they have very specific restrictions on the configuration of the volumes.

- Only one `rootvol` and `swapvol` volume can exist per host.
- `rootvol` and `swapvol` volumes have specific minor device numbers. `rootvol` will be minor device 0 and `swap` will be minor device 1.
- Restricted mirrors of `rootvol` and `swapvol` devices will have "overlay" partitions created for them. An "overlay" partition is one that exactly encompasses the disk space occupied by the restricted mirror. During boot (before the `rootvol` and `swapvol` volumes are fully configured), the default volume configuration uses the overlay partition to access the data on the disk. (See the section entitled"Booting with Root Volumes" on page 1-15.)
- Although it is possible to add a striped mirror to a `rootvol` or `swapvol` device for performance reasons, you cannot stripe any mirrors of `rootvol` or `swapvol` that may be needed for system recovery if the primary mirror fails.
- `rootvol` and `swapvol` cannot be spanned or contain a primary mirror with multiple non-contiguous subdisks.

1.4 Volume Manager Daemons

Two daemons must be running in order for the Volume Manager to work properly:

- `vxconfigd`
- `vxiod`

1.4.1 The Volume Daemon

The volume daemon (`vxconfigd`) is responsible for maintaining a system's configuration in the kernel and on disk. `vxconfigd` must be running before normal operations can be performed.

1.4.1.1 Starting the Volume Daemon

vxconfigd is started by startup scripts during the boot procedure.

To determine whether the volume daemon is enabled, enter the following command:

```
vxdctl mode
```

The following message appears if vxconfigd is both running and enabled:

```
mode: enabled
```

The following message appears if vxconfigd is running, but not enabled:

```
mode: disabled
```

To enable the volume daemon, enter the following:

```
vxconfigd enable
```

The following message appears if vxconfigd is not running:

```
mode: not-running
```

If the latter message appears, start vxconfigd as follows:

```
vxconfigd
```

Once started, vxconfigd automatically becomes a background process. For more information on the vxconfigd daemon, refer to the vxconfigd(1M) and vxdctl(1M) manual pages.

1.4.2 The Volume Extended I/O Daemon

The volume extended I/O daemon (vxiod) serves two purposes:

- It allows for some extended I/O operations without blocking calling processes.
- It allows the virtual disk driver to schedule writes to volumes that have dirty region logging enabled.

If there are volumes with dirty region logging enabled, then there will be multiple vxiod processes running on the system. Volume log I/O vxiod daemons are started by vxconfigd and are killed by the kernel when they are no longer needed.

For more detailed information about vxiod, refer to the vxiod (1M) manual page.

1.4.2.1 Starting the Volume Extended I/O Daemon

vxiod daemons are started at system boot time. There are typically 2 vxiod daemons running at all times. Rebooting after your initial installation should start vxiod.

Verify that vxiod is running by typing the following command:

```
vxiod
```

If any vxiod daemons are running, the following should be displayed:

```
2 volume I/O daemons running
```

where 2 is the number of vxiod daemons currently running.

If no vxiod daemons are currently running, start some by entering the command:

```
vxiod set 2
```

where 2 may be substituted by the desired number of vxiod daemons.

1.5 Volume Manager Interfaces

Volume Manager objects created by one interface are fully inter-operable and compatible with those created by the other interfaces.

The Volume Manager supports the following user interfaces:

- Visual Administrator — The Visual Administrator is a graphical user interface to the Volume Manager. The Visual Administrator provides visual elements such as icons, menus, and forms to ease the task of manipulating Volume Manager objects. In addition, the Visual Administrator acts as an interface to some common file system operations.
- Command Line Interface — The Volume Manger command set consists of a number of comprehensive commands that range from simple commands requiring minimal user input to complex commands requiring detailed user input. Many of the VxVM commands require a thorough understanding of VxVM concepts.
- Volume Manager Support Operations — The Volume Manager Support Operations interface (`vxdiskadm`) provides a menu-driven interface for performing disk and volume administration functions.

VxVM Performance Monitoring 2

2.1 Introduction

Logical volume management, as provided by VxVM, is a powerful tool that can significantly improve overall system performance. This chapter suggests performance management and configuration guidelines that can help the system administrator benefit from the advantages provided by VxVM. It contains information needed to establish performance priorities and describes ways to obtain and use appropriate data.

2.2 Performance Guidelines

VxVM provides flexibility in configuring storage to improve system performance. Two basic strategies are available for optimizing performance:

- assigning data to physical drives in order to evenly balance the I/O load among the available disk drives
- identifying the most-frequently accessed data and increasing access bandwidth to that data through the use of mirroring and striping

2.2.1 Data Assignment

When deciding where to locate file systems, a system administrator typically attempts to balance I/O load among available drives. The effectiveness of this approach may be limited by difficulty in anticipating future usage patterns, as

well as an inability to split file systems across drives. For example, if a single file system receives most of the disk accesses, placing that file system on another drive will only move the bottleneck to another drive.

Since VxVM provides a way for volumes to be split across multiple drives, a finer level of granularity in data placement can be achieved. After measuring actual access patterns, the system administrator can adjust file system placement decisions. Volumes can be reconfigured online after performance patterns have been established or have changed, without adversely impacting volume availability.

2.2.2 Mirroring and Striping

As discussed in Chapter 1, mirroring is a technique for storing multiple copies of data on a system. The use of mirroring improves the chance of data recovery in the event of a system crash or disk failure, and in some cases can be used to improve system performance. Striping is a way of "slicing" data and storing it across multiple devices in order to improve access performance.

Mirroring and striping can be used to achieve a significant improvement in performance when there are multiple I/O streams. Striping can improve serial access when I/O exactly fits across all subdisks in one stripe. Better throughput is achieved because parallel I/O streams can operate concurrently on separate devices.

If the most heavily-accessed file systems and databases can be identified, then significant performance benefits can be realized by striping this "high traffic" data across portions of multiple disks, and thereby increasing access bandwidth to this data. Mirroring heavily-accessed data would not only protect the data from loss due to disk failure, but in many cases could also improve I/O performance.

2.2.3 Mirroring

When properly applied, mirroring can be used to provide continuous data availability by protecting against data loss due to physical media failure. Mirroring can also be used to improve system performance. Unlike striping, however, performance gained through the use of mirroring depends on the read/write ratio of the disk accesses. If the system workload is primarily write-intensive (for example, greater than 30 percent writes), then mirroring can result in somewhat reduced performance.

Since mirroring will most often be used to protect against loss of data due to drive failures, it may sometimes be necessary to use mirroring for write-intensive workloads; in these instances, mirroring can be combined with striping to deliver both high availability and performance.

To provide optimal performance for different types of mirrored volumes, VxVM supports two read policies:

- the round robin read policy, in which read requests to the volume are satisfied in a round-robin manner from all mirrors in the volume
- the preferred read policy, in which read requests are satisfied from one specific mirror (presumably the mirror with the highest performance), unless that mirror has failed, in which case another mirror is accessed.

  For example, in the configuration shown in Figure 2-1, the read policy of the volume labeled “Hot Vol” should be set to `preferred read` to the striped mirror labeled “PL1.” In this way, reads going to PL1 distribute the load across a number of otherwise lightly used disk drives, as opposed to a single disk drive.

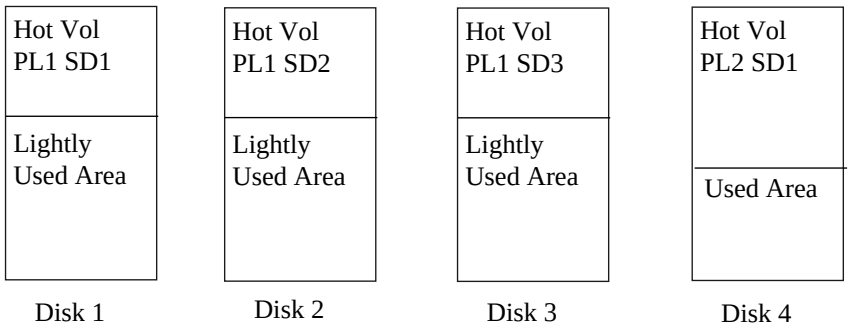

Figure 2-1 Improving System Performance using Mirroring and Striping

To improve performance for read-intensive workloads, up to eight mirrors can be attached to the same volume, although this scenario results in a decrease of effective disk space utilization. Performance can also be improved by striping across half of the available disks to form one mirror and across the other half to form another mirror.

2.2.3.1 Mirroring Guidelines

Follow these guidelines when using mirroring:

- Never place subdisks from different mirrors of a mirrored volume on the same physical disk; this action compromises the availability benefits of mirroring and significantly degrades performance.
- To provide optimum performance improvements through the use of mirroring, at least 70 percent of the physical I/O operations should be reads; a higher percentage of read operations results in a higher benefit of performance. Mirroring may provide no performance increase or result in a decrease of performance in a write-intensive workload environment.

Note – The UNIX operating system implements a file system cache. Since read requests frequently can be satisfied from this cache, the read/write ratio for physical I/O's through the file system can be significantly biased toward writing than the read/write ratio application level.

- Where feasible, use disks attached to different controllers when mirroring or striping. Although most disk controllers support overlapped seeks (allowing seeks to begin on two disks at once), do not configure two mirrors of the same volume on disks attached to a controller that does not support overlapped seeks. This is very important for older controllers or SCSI disks that do not do caching on the drive. It is less important for many newer SCSI disks and controllers.
- If one mirror exhibits superior performance—due to either being striped or concatenated across multiple disks, or because it is located on a much faster device—then the read policy can be set to prefer the "faster" mirror. By default, a volume with one striped plex should be configured with preferred reading of the striped plex.

2.2.3.2 Dirty Region Logging (DRL) Guidelines

Dirty region logging can significantly speed up recovery of mirrored volumes following a system crash. When DRL is enabled, VxVM keeps track of the regions within a volume that have changed as a result of writes to a mirror by maintaining a bitmap and storing this information in a *log subdis*k.

Note – Using dirty region logging may impact system performance in a write-intensive environment where there is not much locality of reference.

Logging subdisks are two-, four-, six-, eight, or ten-block long subdisks that are defined for and added to a mirror that is to become part of a volume that has DRL enabled. They are ignored as far as the usual mirror policies are concerned and are only used to hold the DRL information.

Follow these guidelines when using DRL:

- Make sure that the subdisk that will be used as the log subdisk does not contain necessary data.
- Logging subdisks must be an even number of blocks from two to ten. If a volume is small, a logging subdisk of 2 blocks may be sufficient. The larger the logging subdisk size, the finer the granularity of a dirty region. For extremely large volumes, it is recommended that you create a large (eight or ten block) logging subdisk.
- The log subdisk should not be placed on a heavily-used disk, if at all possible.
- If persistent (non-volatile) expanded storage disks are available, it is desirable to use them for log subdisks.

2.2.4 Striping

Striping can provide increased access bandwidth for a mirror. Striped mirrors exhibit improved access performance for both read and write operations. Where possible, disks attached to different controllers should be used to further increase parallelism.

One disadvantage of striping is that some configuration changes are harder to perform on striped mirrors than on concatenated mirrors. For example, it is not possible to move an individual subdisk of a striped mirror.

While these operations can be performed on concatenated mirrors without “copying through” a mirror, striping offers the advantage that load balancing can be achieved in a much simpler manner. Figure 2-2 is an example of a single file system that has been identified as being a data access bottleneck. This file system was striped across four disks, leaving the remainder of those four disks free for use by less-heavily used file systems.

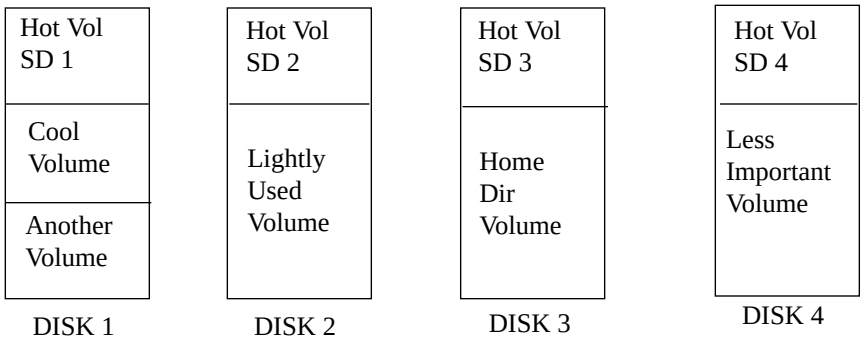

Figure 2-2 Use of Striping for Optimal Data Access

2.2.4.1 Striping Guidelines

Follow these guidelines when using striping:

- Calculate stripe sizes carefully. If it is not feasible to set the stripe width to the track size, use 64 kilobytes for the stripe width.
- Avoid small stripe widths; small stripe widths can result in poor system performance unless its total width exactly matches the size of the I/O requests being done at the application layer.
- Never put more than one column of a striped mirror on the same physical disk.
- Typically, the greater the number of physical disks in the stripe, the greater the improvement in I/O performance; however, this reduces the effective mean time between failures of the volume. If this is an issue, striping can be combined with mirroring to provide a high-performance volume with improved reliability.
- If only one mirror of a mirrored volume is striped, be sure to set the policy of the volume to `preferred read` for the striped mirror. (The default read policy, `select`, does this automatically.)
- When striping is used with mirroring, never place subdisks from one mirror on the same physical disk as subdisks from the other mirror.
- If more than one mirror of a mirrored volume is striped, make sure the stripe width is the same for each striped mirror.

- Where possible, distribute the subdisks of a striped volume across drives connected to different controllers and buses.
- Avoid the use of controllers that do not support overlapped seeks.

The `vxassist` command automatically applies many of these rules when it allocates space for striped plexes in a volume.

2.3 Performance Monitoring

There are two sets of priorities for a system administrator. One set is physical, concerned with the hardware; the other set is logical, concerned with managing the software and its operations.

2.3.1 Performance Priorities

The physical performance characteristics address the balance of the I/O on each drive and the concentration of the I/O within a drive to minimize seek time. Based on monitored results, it may be necessary to move subdisks around to balance the disks.

The logical priorities involve software operations and how they are managed. Based on monitoring, certain volumes may be mirrored (multimirrored) or striped to improve their performance. Overall throughput may be sacrificed to improve the performance of critical volumes. Only the system administrator can decide what is important on a system and what tradeoffs make sense.

2.3.2 Getting Performance Data

VxVM provides two types of performance information: I/O statistics and I/O traces. Each type can help in performance monitoring. I/O statistics are retrieved using the `vxstat` utility, and I/O tracing can be retrieved using the `vxtrace` utility. A brief discussion of each of these utilities is included in this chapter.

2.3.2.1 Obtaining I/O Statistics

The `vxstat` utility provides access to information for activity on volumes, mirrors, subdisks, and disks under VxVM control. `vxstat` reports statistics that reflect the activity levels of VxVM objects since boot time. Statistics for a

specific VxVM object or all objects can be displayed at one time. A disk group can also be specified, in which case statistics for objects in that disk group only will be displayed; if no disk group is specified, rootdg is assumed.

The amount of information displayed depends on what options are specified to `vxstat`. For detailed information on available options, refer to the `vxstat(1M)` manual page.

VxVM records the following three I/O statistics:

- a count of operations
- the number of blocks transferred (one operation could involve more than one block)
- the total active time

VxVM records the preceding three pieces of information for logical I/Os: reads, writes, atomic copies, verified reads, verified writes, mirror reads, and mirror writes for each volume. As a result, one write to a two-mirror volume results in at least five operations: one for each mirror, one for each subdisk, and one for the volume. Similarly, one read that spans two subdisks shows at least four reads—one read for each subdisk, one for the mirror, and one for the volume.

VxVM also maintains other statistical data. For each mirror, read failures and write failures that appear are maintained. For volumes, corrected read failures and write failures accompany the read failures and write failures.

`vxstat` is also capable of resetting the statistics information to zero. Use the command `vxstat -r` to clear all statistics. This can be done for all objects or for only those objects that are specified. Resetting just prior to a particular operation makes it possible to measure the impact of that particular operation afterwards.

The following is an example of `vxstat` output:

```
                 OPERATIONS          BLOCKS          AVG TIME(ms)
TYP NAME       READ    WRITE     READ     WRITE     READ     WRITE
vol blop          0        0        0         0      0.0       0.0
vol foobarvol     0        0        0         0      0.0       0.0
vol rootvol   73017   181735   718528   1114227     26.8      27.9
vol swapvol   13197    20252   105569    162009     25.8     397.0
vol testvol       0        0        0         0      0.0       0.0
```

2.3.2.2 Tracing I/O

The `vxtrace` command is used to trace operations on volumes. Through the `vxtrace` utility, the system administrator can set I/O tracing masks against a group of volumes or to the system as a whole. The `vxtrace` utility can then be used to display ongoing I/O operations relative to the masks. Tracing can be applied to volumes, plexes, subdisks, and physical disks. Each separate user of the I/O tracing can specify the how the desired trace mask is set, independent of all other users. For additional information, refer to the `vxtrace(1M)` manual page.

2.3.3 Using Performance Data

Once performance data has been gathered, it can be used to determine an optimum system configuration in order to make the most efficient use of system resources. The following sections provide an overview of how this data can be used.

2.3.3.1 Using I/O Statistics

Examination of the I/O statistics may suggest reconfiguration. There are two primary statistics to look at: volume I/O activity and disk I/O activity.

Before obtaining statistics, consider clearing (resetting) all existing statistics. Use the command `vxstat -r` to clear all statistics. Clearing statistics eliminates any differences between volumes or disks that might appear due to volumes being created, and also removes statistics from booting (which are not normally of interest).

After clearing the statistics, let the system run for a while and then display the accumulated statistics. Try to let it run during typical system activity. In order to measure the effect of a particular application or workload, it should be run specifically. When monitoring a system that is used for multiple purposes, try not to exercise any one application more than it would be exercised normally. When monitoring a time-sharing system with many users, try to let statistics accumulate during normal use for several hours during the day.

To display volume statistics, use the command `vxstat` with no arguments. This might display a list such as:

```
                OPERATIONS          BLOCKS          AVG TIME(ms)
TYP NAME        READ   WRITE        READ     WRITE      READ    WRITE
vol archive      865     807        5722      3809      32.5     24.0
vol home        2980    5287        6504     10550      37.7    221.1
vol local      49477   49230      507892    204975      28.5     33.5
vol rootvol   102906  342664     1085520   1962946      28.1     25.6
vol src        79174   23603      425472    139302      22.4     30.9
vol swapvol    22751   32364      182001    258905      25.3    323.2
```

To display disk statistics, use the command `vxstat -d`. This might display a list such as:

```
                OPERATIONS          BLOCKS          AVG TIME(ms)
TYP NAME        READ   WRITE        READ     WRITE      READ    WRITE
dm disk01      40473  174045      455898    951379      29.5     35.4
dm disk02      32668   16873      470337    351351      35.2    102.9
dm disk03      55249   60043      780779    731979      35.3     61.2
dm disk04      11909   13745      114508    128605      25.0     30.7
```

Look for volumes with an unusually large number of operations or excessive read or write times. It is normal for `rootvol` to have a large amount of activity relative to other volumes. However, `rootvol` and `swapvol` must be stored on the same disk and must each have single-subdisk plexes.

Note – Do not move `rootvol` or `swapvol` from the boot disks in an attempt to improve performance. Doing so may leave the system unbootable.

If extra disks are available, try to avoid using the boot disks for volumes other than `rootvol` and `swapvol`. If it is necessary to store more volumes on the boot disks, look for volumes that are relatively unused (by looking at the I/O counts). For example, based on the previous I/O statistics, it is reasonable to move the volume archive onto the boot disks.

To move the volume archive onto the boot disk (disk01 here), identify which disk(s) it is on using `vxprint -tvh` archive. This might yield the output:

```
V   NAME         USETYPE     KSTATE   STATE     LENGTH   READPOL    PREFPLEX
PL  NAME         VOLUME      KSTATE   STATE     LENGTH   LAYOUT     NCOL/WDTH MODE
SD  NAME         PLEX        PLOFFS   DISKOFFS  LENGTH   [COL/]OFF  FLAGS
v   archive      fsgen       ENABLED  ACTIVE    204800   SELECT     -
pl  archive-01   archive     ENABLED  ACTIVE    204800   CONCAT     -         RW
sd  disk03-03    archive-01  0        409600    204800   0          c1t2d0s0
```

Looking at the associated subdisks indicates that the archive volume is on disk disk03. To move the volume off disk03, use the command:

```
vxassist move archive !disk03 dest_disk
```

where *dest_disk* is the disk you want to move the volume to. It is not necessary to specify a *dest_disk*. If you do not, the volume will be moved to any available disk with enough room to contain the volume.

For C-shell users, use:

```
vxassist move archive \!disk03 disk01
```

This command indicates that the volume should be reorganized such that no part is on disk03, and that any parts to be moved should be moved to disk01.

Note – The easiest way to move pieces of volumes between disks is to use the Volume manager Visual Administrator. If the Volume Manager Visual Administrator is available on the system, it may be preferable to use it instead of command-line utilities.

If there are two busy volumes (other than the root volume), try to move them so that each is on a different disk, if at all possible.

If there is one volume that is particularly busy (especially if it has unusually large average read or write times), consider striping the volume (or splitting the volume into multiple pieces, with each piece on a different disk). Converting a volume to use striping requires sufficient free space to store an

extra copy of the volume. To convert to striping, create a striped mirror of the volume and then remove the old mirror. For example, to stripe the volume archive across disks disk02 and disk04, use:

```
vxassist mirror archive layout=stripe disk02 disk04
vxplex -o rm dis archive-01
```

After reorganizing any particularly busy volumes, check the disk statistics. If some volumes have been reorganized, clear statistics first and then accumulate statistics for a reasonable period of time.

If some disks appear to be excessively used (or have particularly long read or write times), it may be wise to reconfigure some volumes. If there are two relatively busy volumes on a disk, consider moving them closer together to reduce seek times on the disk. If there are too many relatively busy volumes on one disk, try to move them to a disk that is less busy.

Use I/O tracing (or perhaps subdisk statistics) to determine whether volumes have excessive activity in particular regions of the volume. If such regions can be identified, try to split the subdisks in the volume and to move those regions to a less busy disk.

Note – File systems and databases typically shift their use of allocated space over time, so this position-specific information on a volume is often not useful. For databases, it may be possible to identify the space used by a particularly busy index or table. If these can be identified, they are reasonable candidates for moving to non-busy disks.

Caution – Striping a volume, or splitting a volume across multiple disks, increases the chance that a disk failure will result in failure of that volume. For example, if five volumes are striped across the same five disks, then failure of any one of the five disks will require that all five volumes be restored from a backup. If each volume were on a separate disk, only one volume would have to be restored. Use mirroring to substantially reduce the chances that a single disk failure will result in failure of a large number of volumes.

Examining the ratio of reads and writes helps to identify volumes that can be mirrored to improve their performance. If the read-to-write ratio is high, mirroring could increase performance as well as reliability. The ratio of reads to writes where mirroring can improve performance depends greatly on the

disks, the disk controller, whether multiple controllers can be used, and the speed of the system bus. If a particularly busy volume has a ratio of reads to writes is as high as 5:1, it is likely that mirroring can dramatically improve performance of that volume.

2.3.3.2 Using I/O Tracing

I/O statistics provide the data for basic performance analysis; I/O traces serve for more detailed analysis. With an I/O trace, focus is narrowed to obtain an event trace for a specific workload. Exactly where a hot spot is, how big it is, and which application is causing it can be identified.

Using data from I/O traces, real work loads on disks can be simulated and the results traced. By using these statistics, the system administrator can anticipate system limitations and plan for additional resources.

Disks and Disk Groups 3

3.1 Introduction

This chapter describes the Volume Manager operations that are used to manage disks used by the Volume Manager. Information is included on how to initialize disks for use with the Volume Manager, how to remove disks, and what to do when disks fail. Advanced topics that are discussed include creating disk groups, moving disk groups between systems, and using special-purpose devices (such as RAM disks).

3.2 Standard Disk Devices

There are two classes of disk devices that can be used with the Volume Manager: standard devices and certain special devices. Special devices will be described later in this chapter.

All standard disk devices are defined by a controller number (between 0 and 7), a target address (also between 0 and 6), and a logical unit number (0).

Up to 16 `vtoc` partitions (also called slices) can be created on the physical disk on Solaris 2.x. These partitions are named, in order, 0 through 15. Partition 2 is reserved to indicate the entire disk.

On standard devices, the Volume Manager creates and manages `vtoc` partitions itself. It creates two partitions on each physical disk: a small partition in which it keeps its disk label and other administrative data and a large partition that covers the remainder of the disk. A symbolic name (the *disk*

name or *disk media name*) can be established to refer to a disk that is managed by the Volume Manager (for example: `disk01`, `disk02`). The two partitions are called the public region partition and the private region partition, and are used to store subdisks and configuration information, respectively.

A partition is addressed through a physical address, generally referred to as the *device name* or *devname*, which is comprised of four elements as follows:

- `c`*n*—The number of the controller to which the disk drive is attached
- `t`*n* and `d`*n*—The target ID and device number that constitute the address of the disk drive on the controller.
- `s`*n*—The partition number on the disk drive.

An example of a device name is `c0t0d0s2`. By convention, `s2` is used to address all application accessible space on a disk drive. The physical drive is identified by `c`*n*`t`*n*`d`*n*.

The boot partition (which contains the root file system and is used when booting the system) is usually identified to the Volume Manager by the device name `c0t0d0s2`.

The Volume Manager Support Operations (`vxdiskadm`), and `vxdiskadd` utilities take device names without the s2 suffix. For example, to name the second disk attached to the first controller to `vxdiskadd`, use the name `c0t1d0`.

3.3 Disk Groups

Disks are organized by the Volume Manager into disk groups. A disk group is a named collection of disks that share a common configuration. Volumes are created within a disk group and are restricted to using disks within that disk group.

When a disk is added to a disk group it is given a name, such as `disk02`. This name can be used to identify a disk for volume operations, such as volume creation or mirroring. This name relates directly to the physical disk. If a physical disk is moved to a different target address or to a different controller, the name `disk02` will continue to refer to it. Disks can be replaced by first associating a different physical disk with the name of the disk to be replaced and then recovering any volume data that was stored on the original disk (from mirrors or backup copies).

A system with the Volume Manager installed will always have the disk group `rootdg`. By default, operations are directed to this diskgroup. Most systems do not need to use more than one disk group.

Disks do not have to be added to disk groups. They can be initialized and left unadded. This is useful for disk replacement. A disk that is initialized but not added to a disk group can be used immediately as a replacement for another disk that fails.

3.4 Disk and Disk Group Utilities

The Volume Manager provides three interfaces that can be used to manage disks with the Volume Manager:

- the Volume Manager Visual Administrator
- a set of command-line utilities
- a simple menuing interface defined by the `vxdiskadm` utility.

This chapter describes uses for the `vxdiskadm` utility but does not describe its use in detail (refer to the *VERITAS Volume Manager User's Guide* for more information on `vxdiskadm`). Utilities described in this chapter include:

- `vxdiskadm`—The Volume Manager Support Operations menu. This utility provides a menu of disk operations. Each entry in the main menu leads you through a particular operation by providing you with information and asking you questions. Default answers are provided for many questions so that common answers can be selected easily.
- `vxdiskadd`—This utility is used to add standard disks to the Volume Manager. `vxdiskadd` leads you through the process of initializing a new disk by displaying information and asking questions.

Note – You may occasionally see a disk driver error message when adding a disk using `vxdiskadd`. This message does not affect `vxdiskadd`, and can be ignored.

- `vxdisk`—This is the command-line utility for administering disk devices. `vxdisk` is used to define special disk devices, to initialize information stored on disks that the Volume Manager uses to identify and manage disks, and to perform additional special operations. See the `vxdisk(1M)` man page for complete information on how to use `vxdisk`.

- vxdg—This is the command-line utility for operating on disk groups. This can be used to create new disk groups, to add and remove disks from disk groups, and to enable (import) or disable (deport) access to disk groups. See the vxdg(1M) man page for complete information on how to use vxdg.

Note – The vxdiskadd utility, and most vxdiskadm operations, can be used only with standard disk devices.

3.5 Using Disks

This section describes some of the basic disk administration options that are available with the Volume Manager.

3.5.1 Initializing and Adding Disks

There are two levels of initialization for disks in the Volume Manager. The lowest level is formatting of the disk media itself; the second level involves storing of identification and configuration information on the disk for use by the Volume Manager. Formatting the disk media must be done outside of the Volume Manager. Volume Manager operations are provided that step through the last two levels of disk initialization.

A fully initialized disk can be added to a disk group, used to replace a previously failed disk, or used to create a new disk group. These topics are discussed later in this chapter.

In order to perform the first initialization phase, disks can be formatted in Solaris 2.x using the format utility. To do a media format of any disk use the interactive command:

```
format
```

Note – SCSI disks usually come preformatted, the format command typically is needed only if the format becomes severely damaged.

The next disk initialization phase is accomplished by using the vxdiskadm menus or vxdiskadd. For example, to initialize the second disk on the first controller use the command:

```
vxdiskadd c0t1d0
```

This will examine your disk to determine whether it has been initialized already and will ask questions depending on what it finds. `vxdiskadd` will check for disks that can be encapsulated (encapsulation is described in a later section), for disks that have already been added to the Volume Manager, and for a number of less likely conditions.

Note – If you are adding an uninitialized disk, warning and error messages may be displayed on the console during `vxdiskadd`.

Ignore these messages. These messages should not appear after the disk has been fully initialized.

If the disk is uninitialized, or if you choose to reinitialize the disk, you will be prompted with the following:

```
You can choose to add this disk to an existing disk group, to
create a new disk group, or you can choose to leave the disk
available for use by future add or replacement operations. To
create a new disk group, select a disk group name that does not
yet exist. To leave the disk available for future use, specify
a disk group name of "none".

Which disk group [<group>,none,list,q,?] (default: rootdg)
```

To add this disk to the default group `rootdg`, press Return. To leave the disk unadded to a disk group (to leave the disk free as a replacement disk), enter `none`. After this, you will be asked to select a name for the disk in the disk group:

```
You must now select a disk name for the disk. This disk name
can be specified to disk removal, move, or replacement
operations. If you move the disk, such as between host bus
adapters, the disk will retain the same disk name, even though
it will be accessed using a different disk device address name.

Enter disk name [<name>,q,?] (default: disk02)
```

Normally, you should select the default answer. The following screen is displayed:

```
Surface analysis may be necessary to locate any damaged blocks
on the disk and to arrange for undamaged blocks to be used as
replacements. This may take a few minutes to half an hour.
You can skip this step, if you wish. However, it is advisable
that you not skip this step unless you are certain that this
disk maintains its own list of damaged and replacement blocks,
and that this list is correct. If you aren't sure, do surface
analysis.

Perform surface analysis? [y,n,q,?] (default: y)
```

Press return to continue. The initialization and adding operation will complete without further questions.

3.5.2 Removing Disks

A disk that contains no subdisks can be removed from its disk group with the command: `vxdg rmdisk` *`diskname`* For example, to remove `disk02`, use:

```
vxdg rmdisk disk02
```

If the disk has subdisks on it when you try to remove it, the following error message is displayed:

```
vxdg:Disk diskname is used by one or more subdisks.
```

Use `-k` to remove device assignment. Using the `-k` option allows you to remove the disk in spite of the presence of subdisks. For more information, see the `vxdg(1m)` manual page.

In some cases, you may want to remove a disk that some subdisks are defined on. For example, you may have three disks on one system and you may want to consolidate all of the volumes onto one disk. If you use `vxdiskadm` to remove a disk, you can choose to move volumes off that disk. To do this, run

vxdiskadm and select item 3, Remove a disk from the main menu. If the disk is used by some subdisks, then a screen resembling the following will be displayed:

```
The following subdisks currently use part of disk disk02:

     home usrvol

Subdisks must be moved from disk02 before it can be removed.

Move subdisks to other disks? [y,n,q,?] (default: n)
```

If you choose y, then all subdisks will be moved off of the disk, if possible. Some subdisks may not be movable. The most common reasons why a subdisk may not be movable are:

- There is not enough space on the remaining disks
- Mirrors or stripe subdisks cannot be allocated on different disks from existing mirrors or striped subdisks in the volume.

If vxdiskadm cannot move some subdisks, you may need to remove some mirrors from some subdisks, to get more space before proceeding with the disk removal operation. See Chapter 4 for information on how to remove volumes and mirrors.

3.6 Repairing Disks

This section discusses how to detect disk failures and how to replace failed disks.

3.6.1 Detecting Failed Disks

If one mirror of a volume encounters a disk I/O failure (for example, because the disk has an uncorrectable format error), the Volume Manager may detach the mirror. If a disk fails completely, the Volume Manager may detach the disk from its disk group. If a mirror is detached, I/O stops on that mirror but continues on the remaining mirrors of the volume. If a disk is detached, all mirrors on the disk are disabled. If there are any unmirrored volumes on a disk when it is detached, those volumes are disabled as well.

If a volume, mirror, or disk is detached by failures, mail is sent to `root` indicating the failed objects. For example, if a disk containing two mirrored volumes fails, you might receive a mail message like:

```
To: root
Subject: Volume Manager failures on host minuet.tango.com

Failures have been detected by the VERITAS Volume Manager on host
minuet.tango.com:

failed plexes:
  home-02
  src-02

No data appears to have been lost. However, you should replace
the drives that have failed.
```

See the `vxsparecheck`(1M) manual page for information on sending the mail to users other than `root`.

Note – If you miss the email, the failure can be seen with the `vxprint` command output or by using the Visual Administrator to look at the status of the disks.

To determine which disks are causing the failures in the above message, run the command:

```
vxstat -sf home-02 src-02
```

This will produce output such as:

```
                          FAILED
TYP NAME                READS    WRITES
sd disk01-04                0         0
sd disk01-06                0         0
sd disk02-03                1         0
sd disk02-04                1         0
```

This display indicates that the failures are on `disk02` (the basename for the displayed subdisks).

Sometimes these errors are caused by cabling failures. You should look at the cables connecting your disks to your system. If there are any obvious problems, correct them and recover the mirrors with the command:

```
vxrecover -b home src
```

This command will start a recovery of the failed mirrors in the background (the command will return before the operation is done). If an error message appears later, or if the mirrors become detached again, and there are no obvious cabling failures, you should replace the disk.

If a disk fails completely, the mail message will list the disks that have failed, all mirrors that use the disk, and all volumes defined on the disk that were disabled because the volumes were not mirrored. For example:

```
To: root
Subject: Volume Manager failures on host minuet.tango.com

Failures have been detected by the VERITAS Volume Manager on host
minuet.tango.com:

failed disks:
  disk02

failed plexes:
  home-02
  src-02
  mkting-01

failed volumes:
  mkting

The contents of failed volumes may be corrupted, and should be
restored from any available backups. To restart one of these
volumes so that you can restore it from backup, replace disks as
appropriate then use the command:

     vxvol -f start <volume-name>

You can then restore or recreate the volume.
```

This message indicates that `disk02` was detached by a failure. When a disk is detached, I/O cannot get to that disk. Mirrors `home-02`, `src-02`, and `mkting-01` were also detached (probably because of the failure of the disk); and the volume `mkting` was disabled.

Again, the problem may be a cabling error. If the problem is not a cabling error then you must replace the disk.

3.6.2 Replacing Disks

Disks that have failed completely (that have been detached by failure) can be replaced by running `vxdiskadm` and selecting item 5, `Replace a failed or removed disk` from the main menu. If there are any initialized but unadded disks, you will be able to select one of those disks as a replacement. Do not choose the old disk drive as a replacement; it may appear in the selection list. If there are no suitable initialized disks, you can choose to initialize a new disk.

If a disk failure caused a volume to be disabled then the volume must be restored from backup after the disk is replaced. To identify volumes that wholly reside on disks that were disabled by a disk failure, use the command:

```
vxinfo
```

Any volumes that are listed as `Unstartable` must be restored from backup. For example, `vxinfo` might display:

```
home          fsgen     Started
mkting        fsgen     Unstartable
src           fsgen     Started
standvol      gen       Started
rootvol       root      Started
swapvol       swap      Started
```

To restart volume `mkting` so that it can be restored from backup, use the command:

```
vxvol -obg -f start mkting
```

The `-obg` option causes any mirrors to be recovered in a background task.

If failures are starting to occur on a disk, but the disk has not yet failed completely, you should replace the disk. This involves two steps: detaching the disk from its disk group and replacing the disk with a new one. To detach the disk, run `vxdiskadm` and select item 4, `Remove a disk for replacement`, from the main menu. If there are initialized disks available as replacements, you can specify the disk as part of this operation. Otherwise, you must specify the replacement disk later by selecting item 5, `Replace a failed or removed disk`, from the main menu.

When you select a disk to remove for replacement, all volumes that will be affected by the operation are displayed. For example, the following output might be displayed:

```
The following volumes will lose mirrors as a result of this
operation:

     lhome src

No data on these volumes will be lost.

The following volumes are in use, and will be disabled as a result
of this operation:

     mkting

Any applications using these volumes will fail future accesses.
These volumes will require restoration from backup.

Are you sure you want do this? [y,n,q,?] (default: n)
```

If any volumes would be disabled, you should quit from `vxdiskadm` and save the volume. Either backup the volume or move the volume off of the disk. To move the volume `mkting` to a disk other than `disk02`, use the command:

```
vxassist move mkting !disk02
```

After the volume is backed up or moved, run `vxdiskadm` again and continue to remove the disk for replacement.

After the disk has been removed for replacement, a replacement disk can be used by specifying item 5, `Replace a failed or removed disk`, from the main menu in `vxdiskadm`.

3.6.3 Hot-Sparing

Hot-sparing is used to automatically recover data when a disk fails. When a disk fails, VxVM looks for the closest spare disk and places the data from the failed disk on the spare disk. The replacement disk device must be initialized and placed on the system as a spare.

When a disk failure is detected, VxVM first attempts to correct the object error that brought the failure to its attention. If VxVM is unable to correct the error, `vxconfigd` is notified. `vxconfigd` then tries to access the disk. If `vxconfigd` cannot access the disk, it considers the disk to have failed and sends you a mail message about the failure.

At the same time, `vxconfigd` notifies the `vxsparecheck`. `vxsparecheck` searches for a suitable replacement for the failed disk. The replacement selection is based upon what disks are available and what VxVM objects reside on the failed disk (for example, if the disk that failed was part of a striped volume, the replacement disk should be on a different controller than the other disks in that volume).

Note – If no spare disks large enough to contain all the data from the failed disk are available, VxVM notifies you about the failure, but no other automatic action is taken.

The disk access record of the failed disk is disassociated from its disk media record. The disk media record is then associated with the disk access record of the replacement disk device. This leaves the disk media record for the new device unassociated and the device's error flag is set. The failed device is also unassociated. The display from a `vxdisk list` will show this disk to have last been attached to the original disk media record.

For information on designating a disk as a spare, see the *Veritas Volume Manager User's Guide*.

3.7 Disk Groups

This section describes some of the disk group administrative options available with the Volume Manager.

3.7.1 Creating a Disk Group

A new disk group can be created on a physical disk using `vxdiskadd`. If using `vxdiskadd`, specify a new disk group name when prompted for a disk group. They can also be created using the operation `vxdg init`. To create a disk group with the `vxdg` utility use the command:

```
vxdg init diskgroupname diskname=devicename
```

For example, to create a disk group named `mktdg` on device `c1t0d0s0`, use the command:

```
vxdg init mktdg mktdg01=c1t0d0
```

The disk device given to `vxdg` must have been initialized already with `vxdiskadd`. The disk must not already be added to a disk group.

3.7.2 Using Disk Groups

Most Volume Manager commands allow a disk group to be specified using the *`-g`* option. For example, to create a volume in disk group *`mktdg`* you could use the command:

```
vxassist -g mktdg make mktvol 50m
```

The volume device for this volume is:

```
/dev/vx/dsk/mktdg/mktvol
```

In many cases the disk group does not have to be specified. Most Volume Manager commands use object names specified on the command line to determine the disk group for the operation. For example, a volume can be created on disk `mktdg01` without specifying the disk group name:

```
vxassist make mktvol 50m mktdg01
```

This works for many commands as long as two disk groups do not have objects with the same name. For example, the Volume Manager allows you to create volumes named mktvol in both rootdg and in mktdg. If you do this, you must add `-g mktdg` to any command where you want to manipulate the volume in the mktdg disk group.

3.7.3 Removing a Disk Group

To remove a disk group, unmount and stop any volumes in the disk group and then run the command:

```
vxdg deport diskgroupname
```

Deporting a disk group does not actually remove the disk group. It disables use of the disk group by the system. However, disks that are in a deported disk group can be reused, reinitialized, or added to other disk groups.

If you do not want to remove a disk group, but do want to reorganize by moving a disk between disk groups, remove the disk from one disk group and add it to the other. For example, to move the physical disk `c0t3d0` (attached with the disk name `disk04`) from disk group `rootdg` and add it to disk group `mktdg`, you could use the commands:

```
vxdg rmdisk disk04
vxdg -g mktdg adddisk mktdg02=c0t3d0
```

This can also be done using `vxdiskadm` by selecting item 3, `Remove a disk`, from the main menu, and then selecting item 1, `Add or initialize a disk`.

3.7.4 Moving Disk Groups Between Systems

An important feature of disk groups is that they can be moved between systems. If all disks in a disk group are moved from one system to another, then the disk group can be used by the second system without having to respecify the configuration.

The following steps are used to move a disk group between systems:

1. On the first system, stop and unmount all volumes in the disk group, stop all volumes, then deport (disable local access to) the disk group with the command:

   ```
   vxdg deport diskgroupname
   ```

2. Then, move all the disks to the second system, online the disk devices with the command:

   ```
   vxdisk online diskdevicename ...
   ```

then import (enable local access to) the disk group on the second system with the command: `vxdg import diskgroupname`

3. After the disk group is imported, start all volumes in the disk group with the command:

```
vxrecover -g diskgroupname -sb
```

You may want to move disks from a system that has crashed. In this case, you will not be able to deport the disk group from the first system. When a disk group is created or imported on a system, that system writes a lock on all disks in the disk group. The purpose of the lock is to ensure that dual-ported disks (disks that can be accessed simultaneously by two systems) will not be used by both systems at the same time. If two systems try to manage the same disks at the same time, configuration information stored on the disk will be corrupted and the disk will become unusable.

If you move disks from a system that has crashed or failed to detect the group before the disk is moved, the locks stored on the disks will remain and must be cleared. The system returns the following error message:

```
vxdg:disk group groupname: import failed: Disk is in use by another
host
```

To clear locks on a specific set of devices, use the command:

```
vxdisk clearimport diskdevicename ...
```

Be careful when using this command on systems that really do have dual-ported disks.

In some cases, you may want to import a disk group when some disks are not available. The `import` operation normally fails if some disks for the disk group cannot be found among the disk drives attached to the system. If the `import` operation fails, one of the following error messages will display:

```
vxdg: Disk group groupname: import failed: Disk for disk group not
found
```

or

```
vxdg: Disk group groupname: import failed: Disk group has no valid
configuration copies
```

If some of the disks in the disk group have failed, you can force the disk group to be imported with the command:

```
vxdg -f import diskgroupname
```

All of these operations can be done using `vxdiskadm`. To deport a disk group in `vxdiskadm` select item 9, `Remove access to (deport) a disk group`. To import a disk group, select item 8, `Enable access to (import) a disk group`. The `vxdiskadm` import operation checks for host import locks and asks if you want to clear any that are found. It also starts volumes in the disk group.

3.8 Using Special Devices

This section discusses special devices used by the Volume Manager to perform administrative tasks.

3.8.1 Using vxdisk for Special Encapsulations

In some cases, you may want to encapsulate a disk (see "Custom Installation" in Chapter 1 of the *VERITAS Volume Manager (VxVM) Installation Guide*) that does not have any space that can be used for the Volume Manager private region partition. The normal disk encapsulation procedure using `vxencap` [see `vxencap(1M)`] requires that some space be available at the beginning or end of the disk for storing Volume Manager identification and configuration information.

The `vxdisk` utility can be used to encapsulate disks that do not have available space. This is done using special types of disk devices, called `nopriv` devices, that do not have private regions. To use this, create a partition on the disk device that maps all parts of the disk that you want to be able to access, then add the partition device for that partition with the command:

```
vxdisk define partition-device type=nopriv
```

Here, *partition-device* is the basename of the device in the `/dev/dsk` directory. For example, to use partition 3 of disk device `c0t4d0`, use the command:

```
vxdisk define c0t4d0s3 type=nopriv
```

To create volumes for other partitions on the disk drive, add the device to a disk group, figure out where those partitions reside within the encapsulation partition, then use `vxassist` to create a volume with that offset and length.

A major drawback with using these special encapsulation partition devices is that the Volume Manager cannot track changes in the address or controller of the disk. Normally, the Volume Manager uses identifying information stored in

the private region on the physical disk to track changes in the location of a physical disk. Since `nopriv` devices do not have private regions and thus have no identifying information stored on the physical disk, this cannot occur.

The best use of special encapsulation partition devices is to encapsulate a disk so that the Volume Manager can be used to move space off of the disk. When space is made available at the beginning or end of the disk, the special partition device can be removed and the disk can then be encapsulated as a standard disk device.

A disk group cannot be formed entirely from `nopriv` devices. This is because `nopriv` devices do not provide space for storing disk group configuration information. Configuration information must be stored on at least one disk in the disk group.

3.8.2 Using `vxdisk` for RAM Disks

Some systems support creation of RAM disks. A RAM disk is a device made from system RAM that looks like a small, simple, disk device. Often, the contents of a RAM disk are erased when the system is rebooted. RAM disks that are erased on reboot defeat the Volume Manager's means of identifying physical disks. This is because information stored on the physical disks is used to identify the disk.

`nopriv` devices have a special feature to support RAM disks: a *volatile* option which indicates to the Volume Manager that the device contents do not survive reboots. Volatile devices are handled specially on system startup. If a volume is mirrored, mirrors made from volatile devices are always recovered by copying data from non-volatile mirrors.

To use a RAM disk, a device node must be created for the disk in the `/dev/vx/dsk` and `/dev/vx/rdsk` directories, for example `/dev/vx/dsk/ramd0` and `/dev/vx/rdsk/ramd0`. To define the RAM disk device to the Volume Manager use the command:

```
vxdisk define ramd0 type=nopriv volatile
```

Normally, the Volume Manager will not start volumes that are formed entirely from mirrors with volatile subdisks. This is because there is no mirror that is guaranteed to contain the most recent volume contents.

Sometimes, RAM disks are used in situations where all contents are recreated after reboot. In these situations, you can force volumes formed from RAM disks to be started at reboot using the command:

```
vxvol set startopts=norecov volumename
```

This option can be used only with `gen`-type volumes. See `vxvol`(1M) for more information on the `vxvol set` operation and the `norecov` option.

Volume Administration 4

4.1 VxVM Utilities

This chapter describes the Volume Manager utilities that are used to maintain a system configuration under the control of VxVM. In addition to descriptions of the utilities, information is included about how to manipulate the configuration and create, remove, and maintain VxVM records. Information is presented in the form of both quick references for each utility and more detailed descriptions on how to perform particular operations using the utilities.

Note – Most of the VxVM utilities can only be used by privileged users.

4.2 Individual Utility Descriptions

The following sections describe VxVM utilities that are most commonly used to perform system administration and maintenance functions. This section also describes the function of each command. Utility-specific examples are included in later sections. Detailed information for each of these utilities can be found in their respective manual pages.

4.2.1 *Using* `vxassist`

The `vxassist` command provides a convenient approach to volume management. `vxassist` acts as an automated one-step interface to Volume Manager operations. Unlike most other VxVM commands, `vxassist` does not expect a thorough understanding of Volume Manager concepts by its users. `vxassist` is capable of accomplishing alone most tasks that would otherwise require the use of a sequence of several other VxVM utilities. `vxassist` automatically takes care of all underlying and related operations that would otherwise need to be performed manually by the user (in the form of other commands).

`vxassist` does the following:

- finds space for and creates volumes
- finds space for and creates mirrors for existing volumes
- finds space for and extends existing volumes
- shrinks existing volumes and returns unused space
- provides facilities for the on-line backup of existing volumes

For detailed information about how to use `vxassist`, refer to the `vxassist` manual page.

4.2.2 *Manipulating the Configuration Databases and the Volume Configuration Daemon With vxdctl*

The volume configuration daemon (`vxconfigd`) is the interface between the other VxVM utilities and the kernel `vxconfig` device driver. The `volconfig` device is a special device file created by the Volume Manager that interacts with `vxdctl` to make system configuration changes.

Some `vxdctl` operations involve modifications to the `vxboot` file, which indicates the locations of root configuration copies.

The `vxdctl` utility is the interface to `vxconfigd` and is used for:

- performing administrative tasks related to the state of the daemon
- managing boot information and various aspects of the VxVM root configuration initialization

- manipulating the contents of the `vxboot` file, which contains a list of disks containing root configuration databases.

For detailed information about how to use `vxdctl` refer to the `vxdctl` manual page.

4.2.3 Removing and Modifying Entities With vxedit

The `vxedit` utility has two functions:

1. `vxedit` allows the system administrator to modify certain records in the volume management databases. Only fields that are not volume usage-type-dependent can be modified.

   For information about volume usage types, see "Displaying Installed Usage Types" in Chapter 2.

2. `vxedit` can be used to remove or rename VxVM objects.

In general, VxVM objects that are associated are not removable by `vxedit`. This means that `vxedit` cannot remove:

- A subdisk that is associated with a mirror.
- A mirror that is associated with a volume.

Note – Using the recursive suboption (`-r`) to the removal option of the command removes all objects from the specified object downward. In this way, a mirror and its associated subdisks, or a volume and its associated mirrors and their associated subdisks, can be removed by a single invocation of this command.

For detailed information about how to use `vxedit`, refer to the `vxedit` manual page.

4.2.4 Creating VxVM Objects With vxmake

The `vxmake` utility is used to add a new volume, mirror, or subdisk to the set of objects managed by VxVM. `vxmake` adds a new record for that object to the VxVM database. Records can be created entirely from parameters specified on the command line, or they can be created using a description file.

If operands are specified, then the first operand is a keyword that determines the kind of object to be created, the second operand is the name given to that object, and additional operands specify attributes for the object. If no operands are specified on the command line, then a description file is used to specify what records to create.

A *description file* is a file that contains plain text which describes the objects to be created with vxmake. A description file can contain several commands, and can be edited to perform a list of operations. The description file is read from standard input unless the -d(*description file*) is given. The following is a sample description file:

```
#rectyp     #name       #options
sd          3s1-01      dmname=disk03 offset=0 len=20480
sd          4s1-01      dmname=disk04
plex        db-dsk      layout=STRIPE st_width=16k0 sd=3s1-01,4s1-01
sd          mem1-01     dmname=memdisk01 len=640h
                        comment "Hot spot for dbvol"
plex        db-mem      sd=mem1-01:40320
vol         db          use_type=genplex=db-dsk,db-mem
                        read_pol=PREFER0pref_name=memdbplx
                        comment "Uses mem1 for hot spot in last 5m"
```

By default, this description file is read from standard input. However, by using the option, a filename can be specified. For detailed information about how to use vxmake as well as detailed descriptions and definitions of the object-specific fields specified with vxmake, refer to the vxmake manual page.

4.2.5 Correcting Mirror Problems With vxmend

The vxmend utility performs miscellaneous VxVM usage-type-specific operations on volumes, plexes (sometimes known as mirrors), and subdisks. These operations are used to:

- clear utility fields
- change the state of a volume or mirror
- take a volume or mirror off-line

- place a volume or mirror on-line
- perform specialized actions for objects with a particular usage type

`vxmend` is used primarily to escape from a state that was accidentally reached. The off-line and on-line functions are now available with disk-related commands.

For detailed information about how to use `vxmend` refer to the `vxmend` manual page.

4.2.6 Performing Plex Operations With vxplex

The `vxplex` utility performs VxVM operations on plex or on volume-and-plex (mirror) combinations. The first operand is a keyword that determines the specific operation to perform. The remaining operands specify the configuration objects to which the operation is to be applied. The `vxplex` utility can attach a mirror to a volume and detach a mirror from a volume. A detached mirror does not participate in I/O activity to the volume, but remains associated with the volume. A detached mirror is reattached when a volume is next started.

Another capability provided by `vxplex` is dissociation of a mirror from the volume with which it is associated. When a mirror is dissociated, its relationship to the volume is completely broken. At that point, the mirror is available for other uses and can be associated with a different volume. This functionality is useful as part of a backup procedure.

Additionally, `vxplex` provides options to copy the contents of the specified volume onto all of the named mirrors, moving the contents of one mirror onto a new mirror, and `vxplex` allows for other usage-type-dependent operations to be added.

For detailed information about how to use vxplex refer to the vxplex manual page.

4.2.7 Printing Configuration Information With vxprint

The `vxprint` utility provides a flexible method of displaying information from records in VxVM configuration database. This command can be used to display partial or complete information about any or all objects. The format can be hierarchical to clarify relationships or `vxprint` can tailor the output for use by UNIX system utilities such as `awk`, `sed`, or `grep`.

For detailed information about how to use `vxprint`, refer to the `vxprint` manual page.

4.2.8 Performing Subdisk Operations With vxsd

The `vxsd` utility is used to maintain subdisk-mirror associations. `vxsd` can associate a subdisk with a mirror or dissociate a subdisk from its associated mirror, to move the contents of a subdisk to another subdisk, to split one subdisk into two subdisks that occupy the same space as the original, or to join two contiguous subdisks into one.

Note – Users should be aware that some `vxsd` operations can take a considerable amount of time to complete.

For detailed information about how to use `vxsd` refer to the `vxsd` manual page.

4.2.9 Printing Volume Statistics With vxstat

The `vxstat` utility prints statistics information about VxVM objects and block devices under VxVM control. `vxstat` reads the summary statistics from the volume device files in the directory */dev/vx/rvol* and formats them to the standard output. These statistics represent VxVM activity from the time the system was initially booted or from the last time statistics were cleared. If no VxVM object name is specified, then statistics from all volumes in the configuration database are reported.

For detailed information about how to use `vxstat`, refer to the `vxstat` manual page.

4.2.10 Tracing Volume Operations With vxtrace

The vxtrace utility prints formatted event log records and sets event trace masks. A *trace mask* is used to determine which type of events will be tracked, such as I/O events, configuration changes, or I/O errors. These events are then recorded by the volume driver in a *trace file*. vxtrace reads the volume event log device (/dev/vx/event) and writes formatted log entries to standard output. vxtrace prints log entries from all volumes in the database unless a specific volume to trace is named.

The vxtrace utility can also be used to set trace masks as described above for a specific volume. The system default trace mask can also be changed, and will be used as the trace mask for any volumes for which a trace mask has not been set.

For detailed information about how to use vxtrace refer to the vxtrace manual page.

4.2.11 Performing Volume Operations With vxvol

The volume operations that the vxvol utility performs are:

- initializing a volume
- starting a volume
- stopping a volume
- establishing the read policy for a volume

Starting a volume changes its kernel state from DISABLED or DETACHED to ENABLED; stopping a volume changes its state from ENABLED or DETACHED to DISABLED.

One of two read policies can be selected:

- round—prescribes round-robin reads of enabled mirrors;
- prefer—prescribes preferential reads from a specified mirror.

For detailed information about how to use vxvol refer to the vxvol manual page.

4.3 VxVM Operations

Once the software has been installed and properly initialized, VxVM can be used to perform system and configuration management operations on its objects: disks, disk groups, subdisks, mirrors, and volumes.

Disks and disk groups must be initialized and defined to the Volume Manager before volumes can be created. Once disks and disk groups are defined, volumes can be created in either of the following ways:

manually—Volumes can be built in a “bottom-up” style by first creating the volume’s subdisks, then its mirrors, then the volume itself. The new volume and its associated mirrors and subdisks can then be manipulated, as necessary.

automatically—Volumes can be created automatically using the `vxassist` interface. The `vxassist` utility requires only the basic attributes of the desired volume as input and uses this information to create any associated mirrors and subdisks. `vxassist` can also be used to modify existing volumes, in which case it also automatically modifies any underlying or associated objects.

Note – Most of this chapter focuses on how to create and modify volumes using the manual “bottom-up” approach. However, it is generally more convenient to use the automated `vxassist` approach to volume management. For a detailed discussion on how to use `vxassist`, refer to the *VERITAS Volume Manager User’s Guide.*

The following sections contain detailed descriptions of Volume Manager operations, along with examples that illustrate the use of VxVM utilities.

4.3.1 Online Backup Using `vxassist`

VxVM provides the ability to perform snapshot backups of volume devices. This capability is provided through the `vxassist` and other utilities. There are various possible procedures for doing backups, depending upon the requirements for integrity of the volume contents. These procedures have the same starting requirement: a mirror that is large enough to store the complete contents of the volume. The mirror can be larger than necessary, but if a

smaller mirror is used, an incomplete copy results. The recommended approach to volume backup involves the use of the `vxassist` utility. The `vxassist` procedure is convenient and relatively simple.

The `vxassist`, `snapstart`, `snapwait`, and `snapshot` operations provide a way to do online backup of volumes with minimal interruption of data change and access activity.

The `snapstart` operation creates a write-only backup mirror which gets attached to and synchronized with the volume. When synchronized with the volume, the backup mirror is ready to be used as a snapshot mirror. The end of the update procedure is signified by the new snapshot mirror changing its state to SNAPDONE. This change can be tracked by `vxassist snapwait` operation, which waits until at least one of the mirrors changes its state to SNAPDONE. If the attach process fails, the snapshot mirror is removed and its space is released.

Once the snapshot mirror is synchronized, it continues being updated until it is detached. The system administrator can then select a convenient time at which to create a snapshot volume as an image of the existing volume. The system administrator can also ask users to refrain from using the system during the brief time required to perform the snapshot (typically less than a minute). The amount of time involved in creating the snapshot mirror is long and indefinite in contrast to the brief amount of time that it takes to create the snapshot volume.

The online backup procedure is completed by running a vxassist snapshot command on a volume with a SNAPDONE mirror. This operation detaches the finished snapshot (which becomes a normal mirror), creates a new normal volume and attaches the snapshot mirror to it. The snapshot then becomes a normal, functioning mirror and the state of the snapshot is set to ACTIVE.

If the snapshot procedure is interrupted, the snapshot mirror is automatically removed when the volume is started.

Follow these steps to perform a complete `vxassist` backup:

1. Create a snapshot mirror for a volume as follows:

```
vxassist snapstart volume_name
```

2. When the `snapstart` operation is complete and the mirror is in a SNAPDONE state, select a convenient time to complete the snapshot operation. Inform users of the upcoming snapshot and warn them to save files and refrain from using the system briefly during that time.

3. Create a snapshot volume that reflects the original volume as follows:

```
vxassist snapshot volume_name temp_volume_name
```

Alternatively, follow these steps to perform a `vxassist` backup with the snapstart portion in the background:

1. Create a snapshot mirror for a volume as follows:

```
vxassist snapstart volume_name
```

2. When the `snapstart` operation is complete and the mirror is in a SNAPDONE state, select a convenient time to complete the `snapshot` operation. Inform users of the upcoming snapshot and warn them to save files and refrain from using the system briefly during that time.

3. Create a snapshot volume that reflects the original volume as follows:

```
vxassist snapshot volume_name temp_volume_name
```

4. Use `fsck` or some utility appropriate to the application running on the volume, to clean the temporary volume's contents. For example:

```
fsck -y /dev/vx/rdsk/temp_volume_name
```

5. Copy the temporary volume to tape, or to some other appropriate backup media.

6. Remove the new volume as follows:

```
vxedit -rf rm temp_volume_name
```

4.4 Subdisk Operations

VxVM volumes are composed of two types of objects, mirrors and subdisks. A *mirror* is composed of a series of subdisks linked together in an address space. A *subdisk* is a portion of a physical disk and is defined by disk media, offset, and length. Subdisks are low-level building blocks of a VxVM configuration. The following sections describe each of the operations that can be performed in relation to subdisks. These subdisk operations are:

- creating a subdisk

- removing a subdisk
- displaying a subdisk
- associating a subdisk
- associating logging subdisks
- dissociating a subdisk
- changing a subdisk
- moving a subdisk
- splitting a subdisk
- joining a subdisk

4.4.1 Creating Subdisks

The command to create VxVM objects is `vxmake`. The steps to create a subdisk include specifying the:

- name of the subdisk
- length of the subdisk
- starting point (offset) of the subdisk within the disk
- disk media name

To create a subdisk, the following command is used:

`volmake sd` *`name disk, offset, len`*

For example, the command line to create a subdisk labeled `disk02-01` that starts at the beginning of disk `disk02` and has a length of 8000 blocks looks like this:

```
vxmake sd disk02-01 disk02,0,8000
```

Note – Commands take sizes in blocks. Adding a suffix changes the unit of measure. A k suffix specifies 1024-byte blocks. To preserve (encapsulate) data that exists on the disk, a mirror and volume must be created to cover that data.

4.4.2 Removing Subdisks

Subdisks are generally removed when making changes to the system configuration. To remove a subdisk, use the following command:

```
vxedit rm subdisk_name
```

For example, the command line to remove a subdisk labeled `disk02-01` looks like this:

```
vxedit rm disk02-01
```

4.4.3 Displaying Subdisks

The `vxprint` utility displays information about VxVM objects. To display general information for all subdisks, use the following command:

```
vxprint -st
```

The `-s` option instructs to get information about subdisks. The `-t` option prints a single-line output record that depends on the type of object being listed.

To display complete information about a particular subdisk, use the following command:

```
vxprint -l subdisk_name
```

For example, the command line to obtain all database information on a subdisk labeled `disk02-01` looks like this:

```
vxprint -l disk02-01
```

4.4.4 Associating Subdisks

Associating a subdisk with a mirror places the amount of disk space defined by the subdisk at a specific offset within the mirror. In all cases, the entire area that the subdisk fills must not be occupied by any portion of another subdisk. There are several different ways that subdisks can be associated with mirrors, depending on the overall state of the configuration.

If the system administrator has already created all the subdisks needed for a particular mirror, subdisks are associated at mirror creation by using a command similar to the following:

```
vxmake plex home-1 sd=disk02-01,0s02-00,0s02-01
```

This command creates a mirror `home-1` and associates subdisks `disk02-01`, `0s02-00`, and `0s02-01` with the mirror `home-1` during the mirror creation process. Subdisks are associated in order starting at offset 0. Using a command like this one eliminates the need to specify the multiple commands necessary to create the mirror and then associate each of the subdisks with that mirror. In the previous example, the subdisks are associated to the mirror in the order they are listed (after the `sd=`); the disk space defined as `disk02-01` will be first, the disk space of `0s02-00` is second, and `0s02-01` is third.

This method of associating subdisks is convenient during initial configuration. Subdisks can also be associated with a mirror that already exists. One or more subdisks can be associated with an existing mirror as follows:

```
vxsd assoc plex_name sd_name [sd_name2 sd_name3 ...]
```

For example, the command line to associate subdisks labeled `disk02-01`, `0s02-00`, and `0s02-01` with a mirror labeled `home-1` looks like this:

```
vxsd assoc home-1 disk02-01 0s02-00 0s02-01
```

If the mirror is not empty, the new subdisks are added after any subdisks that are already associated with the mirror, unless the `-l` option is specified with the command. The `-l` option provides a way to associate subdisks at a specific offset within the mirror.

The `-l` option is needed in a case where a system administrator has created a sparse mirror for a particular volume, and wishes to make this mirror complete. To make the mirror complete, it is necessary to create a subdisk of exactly the size needed to fill the hole in the sparse mirror, and then associate the subdisk with the mirror by specifying the offset of the beginning of the hole in the mirror. Use the following command to accomplish this task:

```
vxsd -l offset assoc sparse_plex_name exact_size_subdisk
```

Note – The subdisk must be exactly the right size because VxVM does not allow for the space defined by two subdisks to overlap within a single mirror.

4.4.5 Associating Logging Subdisks

Logging subdisks are two, four, six, eight, or ten-block long subdisks that are defined for and added to a mirror that is to become part of a volume using dirty region logging. Dirty region logging is enabled for a volume when the volume has at least two active mirrors that include a logging subdisk. For a

description of dirty region logging, refer to the “Dirty Region Logging” section in Chapter 1 and “Dirty Region Logging Guidelines” in Chapter 2. Logging subdisks are ignored as far as the usual mirror policies are concerned, and are only used to hold the *dirty region log*.

Note – Only one logging subdisk can be associated with a mirror. Because this subdisk is frequently written, care should be taken to position it on a disk that is not heavily used. Placing a logging subdisk on a heavily-used disk can result in degradation of system performance.

To add a logging subdisk to a mirror, use the following command:

```
vxsd aslog mirror subdisk
```

For example, the command line to associate a subdisk labeled disk02-01 with a mirror labeled vol01-02 (which is already associated with volume vol01) looks like this:

```
vxsd aslog vol01-02 disk02-01
```

4.4.6 Dissociating Subdisks

To break an established relationship between a subdisk and the mirror to which it belongs, the subdisk is *dissociated* from the mirror. A subdisk is dissociated when the subdisk is to be removed or used in another mirror. To dissociate a subdisk, use the following command:

```
vxsd dis subdisk_name
```

To dissociate a subdisk labeled disk02-01 from the mirror with which it is currently associated, use the following command:

```
vxsd dis disk02-01
```

Subdisks can also be removed with the command:

```
vxsd -orm dis subdisk_name
```

4.4.7 Changing Subdisk Information

The `vxedit` utility changes information related to subdisks. To change information relating to a subdisk use the following command:

```
vxedit set field=value ... subdisk_name
```

For example, the command line to change the comment field of subdisk labeled `disk02-01` looks like this:

```
vxedit set comment= "New comment" disk02-01
```

The subdisk fields that can be changed using `vxedit` are:

- name
- the `putil[`*n*`]` fields
- the `tutil[`*n*`]` fields
- `len` (only if the subdisk is dissociated)
- comment

Note – Entering data in the `putil` and `tutil` fields prevents the subdisk from being used as part of a mirror, if it is not already.

4.4.8 *Moving Subdisks*

Moving a subdisk copies the disk space contents of a subdisk onto another subdisk. If the subdisk being moved is associated with a mirror, then the data stored on the original subdisk is copied to the new subdisk, the old subdisk is dissociated from the mirror, and the new subdisk is associated with the mirror, at the same offset within the mirror as the source subdisk. To move a subdisk, use the following command:

```
vxsd mv old_subdisk_name new_subdisk_name
```

For the subdisk move operation to perform correctly, the following conditions must be met:

- The subdisks involved must be the same size.
- The subdisk being moved must be part of an active mirror on an active (ENABLED) volume.
- The new subdisk must not be associated with any other mirror.

4.4.9 Splitting Subdisks

Splitting a subdisk divides an existing subdisk into two subdisks. The `-s` option is required to specify the size of the first of the two subdisks that will be created. To split a subdisk, the following command is used:

```
vxsd -s size split sd newsd newsd2
```

where *sd* is the name of the original subdisk, *newsd* is the name of the first of the two subdisks that will be created, and *newsd2* is the name of the second subdisk to be created.

If the existing subdisk is associated with a mirror before the operation, upon completion of the split, both of the resulting subdisks will be associated to the same mirror.

4.4.10 Joining Subdisks

Joining a subdisk combines two or more existing subdisks into one subdisk. To join subdisks, the subdisks must be contiguous on the same disk; if the selected subdisks are associated, they must be associated with the same mirror, and be contiguous in that mirror. The command to join a subdisk is:

```
vxsd join subdisk1 subdisk2 new_subdisk
```

4.5 Plex Operations

Plexes are logical groupings of subdisks that create an area of disk space independent of any physical disk size. Replication (mirroring) of disk data can be accomplished by defining multiple plexes that are linked together into one volume. The replication provided by mirroring (creating multiple plexes containing identical copies of data) prevents data loss in the event of a single-point disk-subsystem failure. Multiple plexes also provide increased data integrity and reliability.

Plex operations include:

- creating a plex
- backup using mirroring
- removing a plex
- associating a plex

- dissociating and removing a plex
- listing all plexes
- displaying plexes
- changing plex attributes
- changing plex status
- moving plexes
- copying plexes

4.5.1 Creating a Plex

Plexes are created by identifying subdisks and associating them to the plex that you want to create. The `vxmake` command creates VxVM objects. To create a plex from existing subdisks, use the following command:

```
vxmake plex plex_name sd=subdisk_name,...
```

For example, the command line to create a plex labeled `vol01-02` using two existing subdisks labeled `disk02-01` and `disk02-02` and looks like this:

```
vxmake plex vol01-02 sd=disk02-01,disk02-02
```

4.5.2 Backup Using Mirroring

If a volume is mirrored, backup can be performed on that volume by taking one of the volume's mirrors offline for a period of time. This eliminates the need for extra disk space for the purpose of backup only. However, it also eliminates redundancy of the volume for the duration of the time needed for the backup.

Perform a backup of a mirrored volume on an active system as follows:

- Ask users to stop activity for a short time, in order to improve the consistency of the backup.
- Detach one of the volume's mirrors (`vol-01`, for this example):

```
vxplex det vol-01
```

  This operation should only take a few seconds. It will leave the device /dev/vx/plex/vol-01 available as an image of the volume frozen at the time of the detach.

- Check the plex device, if necessary:

```
fsck -F vxfs /dev/vx/plex/vol-01
```

- Create a backup using the plex device: vxdump 0 /dev/vx/plex/vol-01
- Reattach the plex to the volume in order to regain redundancy of the volume:

```
vxplex att vol vol-01
```

4.5.3 Associating Plexes

A plex becomes a participating mirror for a volume by associating the plex with the volume. To associate a plex with an existing volume, use the following command:

```
vxplex att volume_name plex_name
```

For example, the command line to associate a plex labeled `vol01-02` with a volume labeled `vol01` looks like this:

```
vxplex att vol01 vol01-02
```

Alternately, if the volume has not been created, a plex (or multiple plexes) can be associated with the volume to be created as part of the volume create command:

```
vxmake -U usetype vol vol_name plex=plex_name1, plex_name2...
```

For example, the command line to create a mirrored, fsgen-type volume labeled `home` and associate two existing mirrors labeled `home-1` and `home-2` looks like this:

```
vxmake -Ufsgen vol home plex=home-1,home-2
```

4.5.4 Dissociating and Removing Plexes

When a plex is no longer needed, it can be removed. Examples of operations that require plex to be removed are:

- providing free disk space
- reducing the number of mirrors in a volume in order to increase the length of another mirror and its associated volume; the plexes and subdisks are removed, then the resulting space can be added to other volumes

- removing a temporary mirror that was created to backup a volume and is no longer required
- changing the layout of a plex from concatenated to striped, or vice versa.

Caution – In order to save the data on a plex that is to be removed, the original configuration of that plex must be known. Several parameters from that configuration, such as stripe width and subdisk ordering, are critical to the construction of a new plex which would contain the same data. Before such a plex is removed, its configuration should be recorded.

A plex can be dissociated from the associated volume and removed with the following command:

```
vxplex -o rm dis plex_name
```

To dissociate and remove a plex labeled `vol01-02` use the following command:

```
vxplex -o rm dis vol01-02
```

This removes the plex `vol01-02` and all associated subdisks.

Note – Without the `-o rm`, the `vxplex` command dissociates the plex and subdisks, but does not remove them. To remove the dissociated plex and subdisks, use the command:

```
vxedit -r rm plex_name
```

4.5.5 Listing All Plexes

Listing plexes helps identify free plexes that can be used for building volumes. Using the `vxprint` utility with the plex (–p) option lists information about all mirrors; the `–t` option prints a single line of information about the mirror. To list free mirrors, use the following command:

```
vxprint -pt
```

4.5.6 Displaying Plexes

To display detailed information about all plexes, use the following command:

```
vxprint -lp
```

To display detailed information about a specific plex, use the following command:

```
vxprint -l plex_name
```

4.5.7 Changing Plex Attributes

The `comment` field and the `putil` and `tutil` fields are used by the utilities after plex creation. `putil` attributes are maintained on reboot; `tutil` fields are temporary and are not retained on reboot. Both `putil` and `tutil` have three uses and are numbered according to those uses. These fields can be modified as needed. VxVM uses the utility fields marked `putil0` and `tutil0`. Other VERITAS products use those marked `putil1` and `tutil1`; those marked `putil2` and `tutil2` are user fields. Table 4-1 details the uses for the `putil` and `tutil` fields.

Table 4-1 The putil[n] and tutil[n] Fields

Field	Description
`putil0`	This utility field is reserved for use by VxVM utilities and is retained on reboot.
putil1	This utility field is reserved for use by high-level utilities such as VxVA and the Administrative Script interface. This field is retained on reboot.
putil2	This utility field is reserved for use by the system administrator or site-specific applications. This field is retained on reboot.
tutil0	This utility field is reserved for use by VxVM utilities and is cleared on reboot.
tutil1	This utility field is reserved for use by high-level utilities such as VxVA and the Administrative Script interface. This field is cleared on reboot.

Table 4-1 The putil[n] and tutil[n] Fields

Field	Description
tutil2	This utility field is reserved for use by the system administrator or site-specific applications. This field is cleared on reboot.

To change mirror attributes, use the following command:

```
vxedit set field=value ... plex_name ...
```

The command:

```
vxedit set comment="my plex" tutil2="u" uid="admin" vol01-02
```

uses `vxedit` to set the following attributes:

- set the comment field (identifying what the plex is used for) to `my_plex`.
- set `tutil2` to `u` to indicate that the subdisk is in use.
- change the user ID to `admin`.

To prevent a particular plex from being associated with a volume, set the putil0 field to a non-null string as specified in the following command:

```
vxedit set putil0="DO-NOT-USE" vol01-02
```

4.5.8 Changing Mirror Status: Detaching and Attaching Plexes

Once a volume has been created and placed online (ENABLED), VxVM provides mechanisms by which plexes can be temporarily disconnected from the volume. This is useful, for example, when the hardware on which the plex resides needs repair or when a volume has been left unstartable and a source plex for the volume revive must be chosen manually.

Resolving a disk or system failure includes taking a volume offline and attaching and detaching its plexes. The two commands used to accomplish disk failure resolution are `vxmend` and `vxplex`.

To take a mirror OFFLINE so that repair or maintenance can be performed on the physical disk containing that plex's subdisks, use the following command:

```
vxmend off plex_name ...
```

If a disk drive suffered a head crash, the system administrator should put all plex that have associated subdisks represented on the affected drive OFFLINE. For example, if plexes `vol01-02` and `vol02pl` had subdisks on a drive to be repaired, use the following command:

```
vxmend off vol01-02 vol02pl
```

This command places vol01-02 and vol02pl in the OFFLINE state, and they remain in that state until explicitly changed.

4.5.8.1 Detaching Plexes

To temporarily detach one plex in a mirrored volume, use the following command:

```
vxplex det plex_name
```

For example, the command line to temporarily detach a plex labeled `vol01-02` and place it in maintenance mode looks like this:

```
vxplex det vol01-02
```

This command temporarily detaches the plex, but maintains the association between the plex and its volume; however, the plex will not be used for I/O. A plex detached with the preceding command will be recovered on a system reboot. The plex state is set to STALE, so that if a `vxvol start` command is run on the appropriate volume (for example, on system reboot), the plex will be revived and made ACTIVE.

When the plex is ready to return as an active part of its volume, follow this procedure:

- If the volume is not ENABLED, start it using

```
vxstart vol_name
```

  If it is unstartable, set one of the mirrors (plexes) to CLEAN using

```
vxmend mirror clean plex_name
```

  and then start the volume.

- If the plex does not yet have a *kernel state* of ENABLED, issue the following command:

```
vxplex att volume_name plex_name ...
```

As with returning an OFFLINE plex to ACTIVE, this command starts a revive of the plexes stated, and when each revive completes, sets the plex state to ACTIVE.

4.5.8.2 Attaching Plexes

When the disk has been repaired or replaced and is again ready for use, the plexes must be put back online (plex state set to ACTIVE).

1. If the volume is currently ENABLED, use the following command:

```
vxplex att volume_name plex_name ...
```

For example, the command line for a plex labeled `vol01-02` on a volume labeled `vol01` looks like this:

```
vxplex att vol01 vol01-02
```

This starts a revive of the plex and, after the revive is complete, sets the plex utility state to ACTIVE.

2. If the volume in not in use (not ENABLED), use the following command:

```
vxmend on plex_name
```

For example, the command line for a plex labeled `vol01-02` looks like this:

```
vxmend on vol01-02
```

In this case, the state of `vol01-02` is set to STALE, so that when the volume is next started, the data on the plex will be revived from the other plex, and incorporated into the volume with its state set to ACTIVE.

If it becomes necessary to manually change the state of a plex, refer to "Volume Recovery." See the `vxmake` and `vxmend` manual pages for more information about these commands.

4.5.9 Moving Plexes

Moving a plex copies the data content from the original plex onto a new plex. In order for a move operation to be successful, the following criteria must be met:

- The old plex must be an active part of an active (ENABLED) volume.
- The new plex should be at least the same size or larger than the old plex.

- The new plex must not be associated with another volume.

The size of the plex has several important implications. If the new plex is smaller, or more sparse, than the original plex, an incomplete copy of the data on the original plex results. If this is the desired action, then the `-o force` option is required. If the new plex is longer, or less sparse, than the original plex, the data that exists on the original plex will be copied onto the new plex. Any area that was not on the original plex, but is represented on the new plex, will be filled from other complete plex associated with the same volume. If the new plex is longer than the volume itself, then the remaining area of the new plex above the size of the volume will not be initialized.

The command to move data from one plex to another is:

```
vxplex mv original_plex new_plex
```

4.5.10 Copying Plexes

This operation copies the contents of a volume onto a specified plex. The volume to be copied must not be enabled. The plex must not be associated with any other volume. To copy a plex, the following command is used:

```
vxplex cp vol_name new_plex
```

After the copy operation is complete, *new_plex* will not be associated with the specified volume *vol_name*. The plex contains a complete copy of the volume data. The plex that is being copied should be the same size or larger than the volume, otherwise an incomplete copy of the data results. For this same reason, *new_plex* also should not be sparse.

4.6 Volume Operations

A *volume* is a collection of from one to eight mirrors and appears as a block device in the `/dev/vol` directory and a character device in the `/dev/rvol` directory. A volume can be used as a partition device.

Volume operations include:

- creating a volume
- removing a volume
- listing all volumes
- displaying volumes

- changing volume attributes
- resizing a volume
- changing volume read policy
- starting and stopping volumes
- listing unstartable volumes
- mirroring an existing volume
- displaying volume mirrors
- recovering a disabled volume
- initializing a volume

The following sections describe how to perform common volume operations. In some cases, either `vxassist` or one or more other commands can be used to accomplish the same task; in such cases, both approaches are described. Detailed descriptions about the commands used to perform volume operations are contained in the VxVM manual pages.

4.6.1 `vxassist` *Command Features*

The `vxassist` command provides a convenient, one-step interface to the Volume Manager and is especially useful for basic and commonly used administrative operations. `vxassist` is capable of automatically finding space for and creating simple volumes or mirrors for existing volumes, resizing volumes, and providing online backup of volumes.

The `vxassist` command is capable of accomplishing alone many tasks that would otherwise require the use of a sequence of several other VxVM utilities. `vxassist` does not conflict with the existing VxVM utilities or preclude their use. Objects created by `vxassist` are compatible and inter-operable with objects created manually using the other VxVM utilities and interfaces, and vice versa.

In general, it is more convenient to use `vxassist` than a series of other VxVM commands. Some of the advantages of using `vxassist` include:

- The use of `vxassist` involves only one step on the part of the user. `vxassist` automatically takes care of all underlying and related operations (such as creating associated subdisks and plexes) that would otherwise need to be performed manually by the user through additional commands.

- The user is required to specify only minimal information to `vxassist`, yet can optionally specify additional parameters to modify its actions.
- `vxassist` operations result in a set of configuration changes that either succeed or fail as a group, rather than individually. Most `vxassist` operations therefore function in such a way that system crashes or other interruptions do not leave intermediate states that need to be cleaned up. If vxassist encounters an error or some other exceptional condition, it will exit without leaving behind partially changed configurations; the system will be left in the same state as it was prior to the attempted `vxassist` operation.

4.6.1.1 *How* `vxassist` *Works*

The `vxassist` utility allows users to create and modify simple volumes. The user specifies the basic volume creation or modification requirements and `vxassist` proceeds to perform all of the necessary underlying tasks.

The amount of information that `vxassist` requires from the user is minimal because `vxassist` obtains most of the information it needs from other sources. `vxassist` obtains information about the existing objects and their layouts from the objects themselves. For operations requiring new disk space, seeks out available disk space and tries to allocate it in the configuration that conforms to the layout specifications and offers the best use of free space.

4.6.1.2 `vxassist` *Defaults*

The `vxassist` invocation is designed to be as simple as possible, while allowing its behavior to be tailored when necessary. `vxassist` uses a set of tunable parameters, which can be specified in defaults files or at the command line. The tunable parameters are defaulted to reasonable values if they are not mentioned anywhere. Any tunables listed on the command line override those specified elsewhere. The tunable parameters are specified as follows:

internal defaults—
The built-in defaults are used when the value for a particular tunable is not specified elsewhere (on the command line or in a defaults file).

system-wide defaults file—
The system-wide defaults file contains default values that may be altered by the system administrator. These values are used for tunables that are not specified on the command line or in the user's defaults file.

user defaults file—
The user can create a personal defaults file. If a personal defaults file exists, the values therein are used for tunables that are not specified on the command line. These values override those in the system-wide defaults file.

command line—
The tunable values specified on the command line override any values specified internally or in defaults files.

4.6.1.3 Defaults File

The default behavior of vxassist is controlled by the tunables specified in the `/etc/default/vxassist` file. The format of the defaults file is a list of `attribute=`*`value`* pairs separated by new lines. These `attribute=`*`value`* pairs are the same as those specified as options on the command line (refer to the section entitled "Options").

The following is a sample `vxassist` defaults file:

```
# VxVM 1.2
# vxassist defaults file. Use '#' for comments

# layout
layout=concat,noncontig,span

# mirroring
nmirror=2
mirror=no

# allocation policies
align=4k
alloc=20m

# striping
stripewidth=64k

# logging
logtype=none

# volume usage type
usetype=fsgen
```

4.6.2 Creating a Volume

Volumes can be created with either `vxassist` or `vxmake`.

The length of a new volume can be specified in sectors, megabytes, or kilobytes. The unit of measure is indicated by adding the appropriate suffix to the length (`s`, `m`, or `k`). If no unit is specified, sectors are assumed.

4.6.2.1 vxassist

The `vxassist` command can be used to create volumes with default settings or with user-specified attributes. `vxassist` automatically creates and attempts to enable new volumes. If the volume fails to be enabled, `vxassist` will attempt to remove it and release the space used to allocate that volume.

To create a simple volume using vxassist and its default settings, use the following command:

```
vxassist make volume_name length
```

For example, the following command creates a volume named voldef with a length of 10 megabytes on any available disk(s):

```
vxassist make voldef 10m
```

Additional parameters can be specified to vxassist to reflect the new volume's attributes. Refer to the vxassist(1m) manual page for details. The following example illustrates the creation of a volume named volzebra that is striped across disk03 and disk04, has the fsgen usage type, and is 10 megabytes long:

```
vxassist -Ufsgen make volzebra 10m layout=stripe disk03 disk04
```

4.6.2.2 vxmake

To create a volume using vxmake, use the following command:

```
vxmake -Uusage_type vol volume_name len=length plex=plex_name,...
```

If you do not specify a length, the volume length will equal the length of the plex to which it is attached. You can select a length (less than or equal to the length of the plex) by specifying a length with the len= parameter. You can also create a volume without attaching a plex to it. You do this by omitting the plex= parameter. In this case, you must specify a length. The volume you create will not be available for use until you attach a plex to it using vxplex att.

Examples of commands for creating a fsgen-type volume called vol01 are:

```
vxmake -Ufsgen vol vol01 len=100000
```

or

```
vxmake vol vol01 use_type=fsgen plex=vol01-01,vol01-2
```

The usage type for a volume can be specified in either of two ways: -Ufsgen or use_type=fsgen. If a length is not specified (for example, len=100000) or associated mirrors are not identified (for example, plex=vol01,vol02), length will be zero.

Instead of specifying parameters on the command line, you can use a `vxmake` description file to create a volume, as well as associated subdisks and mirrors, by using the following command:

```
vxmake -d description_file
```

For detailed information about how to use `vxmake`, and an example of the `vxmake` description files, refer to the "Creating Volume Manager Objects With vxmake" section earlier in this chapter or to the `vxmake` manual page.

4.6.3 Initializing a Volume

During normal system operation, volume and mirror states will be affected by system failures, shutdowns, and possible I/O failures. When a volume is first created, it is necessary to initialize the state of its one or more mirrors according to what the state of the data is on each mirror. Normally, if the user has created the volume using `vxassist` or one of the other higher-level interfaces, the state of the mirrors will be properly set. However, when `vxmake` has been used to create a volume, the states of its mirrors must be set manually before the volume can be made available for use through the `vxvol start` command. The command for setting the state of a volume's mirrors is:

```
vxvol init state volume_name [mirror_name]
```

The *state* variable determines what the initialization does and what condition the volume and mirrors will have after the volume has been initialized. The most common form of manual initialization is setting the state of the volume to CLEAN. The following examples show how to do this for mirrored and non-mirrored volumes. In the simplest case, in which a volume has been created containing only one mirror (plex), the state of the plex is set to CLEAN. This is because there is no need for any synchronization of the data on the disk. Since there is only one plex in the volume, it is not necessary to specify the *mirror_name* argument. The command to set the state of this volume to CLEAN is:

```
vxvol init clean volume_name
```

Under more complicated circumstances, where a newly created volume `vol01` has multiple mirrors associated with it, then one of the mirrors must be chosen to which the other mirrors are synchronized. For instance, if mirror vol01-02 has been created over disk space that contained data that needed to be

accessed through the volume after it is made available, then the following command would ensure that the data is synchronized out to the other mirrors when the volume is started:

```
vxvol init clean vol01 vol01-02
```

This command will set the state of `vol01-02` to CLEAN and the remainder of the mirrors to STALE, so that they will be properly synchronized at the time the volume is made available. Sometimes, the administrator will wish to avoid the initial synchronization of the volume in order to save time, with the predefined knowledge that none of the mirrors contain data that will be the final contents of the volume. Under such a situation, it is possible to temporarily initialize the state of the volume so that the data can be loaded without having to perform a synchronization first. The command to do this is:

```
vxvol init enable volume_name
```

This enables the volume and all its mirrors, but leaves the mirror utility states set to EMPTY. After the entire volume's contents have been restored, both mirrors contain exactly the same data and will not need to be synchronized using the `vxvol start` operation. Such a volume, for example `home1`, could be initialized for use and started at the same time using the following command:

```
vxvol init active home1
```

Warning – It is critical that data on each of the mirrors be exactly the same under these circumstances. Otherwise, the system will be likely to corrupt the data on both mirrors and possible crash the system. if you are not sure that the data is identical, then use the `vxvol init clean` method.

Sometimes it is necessary to remove all existing data from disks before new data is loaded. In this case, you can initialize every byte of the volume to zero by entering:

```
vxvol init zero volume_name
```

4.6.4 Removing Volumes

Removing a volume is the same as removing a physical disk partition on a non-volume-managed system. If the volume has been used, and critical data may still reside on the disk space defined by that volume, the system administrator should make a backup of the data, and take care to ensure that this data is retained.

To remove a volume, use the following command:

```
vxedit rm volume_name
```

or

```
vxedit -rf rm volume_name
```

The -r option indicates recursive removal, which means the removal of all mirrors associated with the volume and all subdisks associated with those mirrors. The -f option forces removal, and is necessary if the volume is enabled.

Warning – The `-r` option of `vxedit` removes multiple objects. Exercise caution when using it.

4.6.5 Displaying Volumes

It is possible to list information related to volumes under VxVM control. This information includes the name of the volume, its usage type, state, length, user and group IDs, and mode. To list information on all volumes, use the following command:

```
vxprint -vt
```

To display detailed information about a specific volume, use the following command:

```
vxprint -l volume_name
```

If no volume is specified, detailed information is given for all volumes by using the following command:

```
vxprint -vl
```

4.6.6 Changing Volume Attributes

Volume attributes such as read policy, error policies, ownership, permissions, and the values in the comment and utility fields for existing volumes can be changed. These attributes are changed whenever the use of the volume or users' needs change.

There are two VxVM commands associated with setting volume attributes:

- The `vxedit` command sets those attributes that are not usage-type-dependent.
- The `vxvol` command sets only those attributes that are usage-type-dependent.

Examples of how to use each of these commands follow:

```
vxvol set field=value0 ... volume_name ...
```

or

```
vxedit set field=value0 ... volume_name ...
```

Table 4-2 details which attributes can be set by each command.

Table 4-2 Setting Volume Attributes

Command	Attribute	Description
vxedit	comment	the comment field
	tutil0, tutil1, tutil2 putil0, putil1,putil2	descriptive string of volume contents
	fstype	string indicating file system type
	writeback	boolean (on/off) specifying read error correction mode
	user	owner of volume
	group	group of volume
	mode	permission mode for volume

Table 4-2 Setting Volume Attributes

Command	Attribute	Description
vxvol	len	numeric length of volume (drl/undef)
	log type	specifier of dirty region logging mode for volume
	log len	length of the dirty region logging log
	start opts	options to be executed to the vxvol `start` operation

Note – Setting volume device permissions and ownership using the `chgrp`, `chown`, and `chmod` commands is ineffective. It is necessary to use `vxedit set` to modify these values.

For example, to change the owner of the group to `susan` and the permissions to read/write for owner, group, and other:

```
vxedit set user=susan group=staff mode=0666 vol01
```

4.6.6.1 *Resizing a Volume*

Resizing a volume is an instance of changing volume attributes that can be handled via either `vxassist` or `vxvol`. Striped volumes cannot be resized.

The new size of a volume can be specified in sectors, megabytes, or kilobytes. The unit of measure is indicated by adding the appropriate suffix to the length (`s`, `m`, or `k`). If no unit is specified, sectors are assumed.

4.6.6.2 `vxassist`

`vxassist` can resize a volume in any of the following ways:

`growto`—increase volume to specified length

`growby`—increase volume by specified amount

`shrinkto`—reduce volume to specified length

`shrinkby`—reduce volume by specified amount

If the volume is increased in size, `vxassist` automatically seeks out available disk space.

To increase a volume to a specified length, use the command:

```
vxassist growto volume_name new_length
```

To increase a volume by a certain amount, use the command:

```
vxassist growby volume_name length_change
```

To reduce a volume to a specified length, use the command:

```
vxassist shrinkto volume_name new_length
```

To reduce a volume by a certain amount, use the command:

```
vxassist shrinkby volume_name length_change
```

Note – When a volume's size is reduced using the `vxvol shrinkby` command, the freed space is not released into the free space pool.

4.6.6.3 *vxvol*

To change the length of a volume using `vxvol set`, use the following command:

```
vxvol set len=value ... volume_name ...
```

For example, to change the length to 100000 sectors, use the following command:

```
vxvol set len=100000 vol01
```

Note – The `vxvol set len` command cannot increase the size of a volume unless the needed space is available in the mirrors of the volume.

4.6.6.4 *Changing Volume Read Policy*

VxVM offers the choice of two read policies; `round` reads each mirror in turn in "round-robin" fashion; `prefer` reads preferentially from a mirror that has been labeled as the preferred mirror. The read policy can be changed from `round` to `prefer` (or vice versa in the case of prefer) or to a different preferred mirror.

The command sets the read policy for a volume. To set a read policy, use one of the following commands:

```
vxvol rdpol round volume_name
```

or

```
vxvol rdpol prefer volume_name preferred_mirror_name
```

For example, the command line to set the read policy for volume `vol01` to a round-robin read looks like this:

```
vxvol rdpol round vol01
```

The command line to set the policy for the same volume to read preferentially from the mirror looks like this:

```
vxvol rdpol prefer vol01 vol01-02
```

4.6.7 Starting and Stopping Volumes

Like mounting and unmounting a file system, starting and stopping a volume affects its availability to the user. Starting a volume changes its state and makes it available for use. Stopping a volume makes it unavailable.

Starting a volume changes the volume state from DISABLED or DETACHED to ENABLED. The success of this operation depends on the ability to enable a volume. If a volume cannot be enabled, it remains in its current state. To start a volume, use the following command:

```
vxrecover -s volume_name ...
```

To start all DISABLED volumes, use the following command:

```
vxrecover -s
```

Stopping a volume changes the volume state from ENABLED or DETACHED to DISABLED. If the command cannot stop it, the volume remains in its current state. To stop a volume, use the following command:

```
vxvol stop volume_name ...
```

For example, the command line to stop a volume labeled `vol01` looks like this:

```
vxvol stop vol01
```

To stop all ENABLED volumes, use the following command:

```
vxvol stopall
```

If all mirrors of the volume become STALE, put the volume in maintenance mode so that the mirrors can be looked at while the volume is DETACHED and determine which mirror to use for reviving the others. To place a volume in maintenance mode, use the following command:

```
vxvol maint volume_name
```

To assist in choosing the revival source mirror, list the unstarted volume and displays its mirrors.

To take mirror `vol01-02` offline, use the following command:

```
vxmend off vol01-02
```

For ENABLED volumes, save a step by using `vxplex att` without first invoking `vxmend on`. This command works on an OFFLINE mirror of an ENABLED volume (designated as `vol01` in the example):

```
vxplex att vol01 vol01-02
```

The `vxmend` utility can change the state of an OFFLINE mirror of a DISABLED volume to STALE, after which a `vxvol start` on the volume would revive the mirror. To put a mirror labeled `vol01-02` in the STALE state, use the following command:

```
vxmend on vol01-02
```

To make other state changes in a mirror or a volume, refer to the subsequent sections on volume recovery options.

4.6.8 Listing Unstartable Volumes

An unstartable volume is likely to be incorrectly configured or has other errors or conditions that prevent it from being started. To display unstartable volumes, use the command `vxinfo`, which displays information on the accessibility and usability of one or more volumes:

```
vxinfo [volume_name]
```

4.6.9 Mirroring Existing Volumes

A mirror can be added to an existing volume. This can be done with the `vxassist` command as follows:

```
vxassist mirror volume_name
```

For example:

```
vxassist mirror voltest
```

creates a mirror of the volume `voltest`.

Another way to mirror an existing volume is by first creating a plex and then associating it to a volume, using the following commands:

```
vxmake plex plex_name sd=subdisk_name ...
vxplex att volume_name plex_name
```

4.6.10 Displaying Mirrors Within a Volume

To limit the display of `vxprint` to a single object, specify the object name after the `vxprint` command. To display the mirrors for a volume labeled `vol01` use the following command:

```
vxprint -tv vol01
```

This command displays the volume, its mirrors, and the subdisks in those mirrors.

You can also display the mirrors for `vol01` with the command:

```
vxprint -e 'assoc= "vol01"'
```

This displays only the mirrors associated with `vol01`.

4.6.11 Volume Recovery

If a system crash or an I/O error corrupts one or more mirrors of a volume and no mirror is CLEAN or ACTIVE, mark one of the mirrors CLEAN and instruct the system to use that mirror as the source for reviving the others. To place a mirror in a CLEAN state, use the following command:

```
vxmend fix clean mirror_name
```

For example, the command line to place one mirror labeled in the CLEAN state looks like this:

```
vxmend fix clean vol01-02
```

For detailed information about how to use `vxmend` refer to the `vxmend` manual page.

Volume Manager Error Messages A

A.1 Introduction

VxVM is fault-tolerant and resolves most problems without system administrator intervention. If the volume configuration daemon (`vxconfigd`) recognizes what actions are necessary, it will queue up the transactions that are required. Volume Manager transactions are based on a two-phase commit scheme. It is a method for providing atomic changes of system configurations; either a transaction completes fully or the system appears as though the transaction was never attempted. When `vxconfigd` is unable to recognize and fix system problems, the system administrator needs to handle the task of problem solving.

The following sections cover the majority of informational, failure, and error messages displayed by `vxconfigd` and the kernel driver. These messages are displayed on the console. These sections include some errors that are infrequently encountered and difficult to troubleshoot. Clarifications are included to elaborate on the situation or problem that may have generated a particular message. Wherever possible, a recovery procedure (user action) is provided to locate and correct potential problems.

Should it be necessary to contact your customer support organization, these messages are numbered for ease of reference.

A.1.1 Volume Configuration Daemon Error Messages

The following are the error messages associated with the volume daemon.

A.1.1.1 -r must be followed by 'reset'

```
-r must be followed by 'reset'
```

♦ **Clarification**

This message is caused by a usage error.

♦ **User Action**

Correct the usage and try again.

A.1.1.2 prefix too long

```
-x argument: prefix too long
```

♦ **Clarification**

The stub-mode device path prefix name supplied exceeded the maximum of 32 characters.

♦ **User Action**

Select an alternate path for device files and retry the command.

A.1.1.3 invalid debug string

```
-x string: invalid debug string
```

♦ **Clarification**

An unknown argument string was given to the `-x` option to `vxconfigd`.

♦ **User Action**

Select a valid string from the manual page for `vxconfigd` and try again.

A.1.1.4 Usage: vxconfigd [-dkf] [-r reset] [-m mode] [-x level]

```
Usage: vxconfigd [-dkf] [-r reset] [-m mode] [-x level]

For detailed help use: vxconfigd help
```

♦ **Clarification**

vxconfigd was invoked with an invalid set of arguments.

♦ **User Action**

Correct the usage and try again or type vxconfigd help for more help.

A.1.1.5 Usage: vxconfigd [-dkf] [-r reset] [-m mode] [-x level]

```
Usage: vxconfigd [-dkf] [-r reset] [-m mode] [-x level]
Recognized options:
-d       set initial mode to disabled for transactions
-k       kill the existing configuration daemon process
-f     operate in foreground; default is to operate in background
-r reset reset kernel state; requires 'reset' option argument
-m mode  set vxconfigd's operating mode
         modes: disable, enable, bootload, bootstart
-x debug set debugging level to <debug>, 0 turns off debugging
-R file  set filename for client request rendezvous
-D file  set filename for client diag request rendezvous
```

♦ **Clarification**

This is the full usage message that results from entering vxconfigd help.

♦ **User Action**

Correct the usage and try again.

A.1.1.6 /dev/vx/volevent:cannot open /dev/vx/config: Cannot kill existing daemon

```
vxvm:vxconfigd: Error: /dev/vx/volevent: error_message
vxvm:vxconfigd: Error: cannot open /dev/vx/config: error_message
vxvm:vxconfigd: Error: Cannot kill existing
daemon,pid=process_id
```

♦ **Clarification**

An attempt to kill an existing `vxconfigd` process with a SIGKILL signal has failed. This might be due to the process being in an unkillable kernel state perhaps because of a hung I/O or a missing I/O interrupt.

♦ **User Action**

Try typing `cat /dev/vx/osm` to see if any other messages have been output to the console device. If possible, use `crash` to determine the state of the process. If the process is asleep waiting for an I/O completion, then any disk driver error messages that have occurred might point to the solution. Failing this, a reboot is recommended.

A.1.1.7 /dev/vx/iod: VOL_LOGIOD_KILL failed

```
vxvm:vxconfigd: Error: /dev/vx/iod: VOL_LOGIOD_KILL failed
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.8 All transactions are disabled

```
vxvm:vxconfigd: Error: All transactions are disabled
```

♦ **Clarification**

This message may appear with the message `Disk group disabled by errors` if the disk group to be disabled is the root disk group. The continued use of the system could be dangerous since any configuration changes required (including error handling cases) could cause the loss of ability to perform I/O to a volume. Since this includes the root volume, this situation could, if uncorrected, cause the system to hang.

♦ **User Action**

This is a fatal error. All copies of the bootable root disk have failed. Recovery from this situation will require booting from floppy or from a disk unconnected with VxVM. It may then be necessary to remove the VxVM rootable disk configuration by using the `vxunroot` command. See the VxVM install guide for details. Once this has been achieved, the root disk group can be reinitialized to reestablish the database and log areas.

A.1.1.9 Cannot get all disk groups from the kernel

```
vxvm:vxconfigd: Error: Cannot get all disk groups from the kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a re-install of the VxVM package. Failing this, contact Customer Support.

A.1.1.10 Cannot get all disks from the kernel

```
vxvm:vxconfigd: Error: Cannot get all disks from the kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a re-install of the VxVM package. Failing this, contact Customer Support.

A.1.1.11 Cannot get kernel transaction state

```
vxvm:vxconfigd: Error: Cannot get kernel transaction state
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.12 Cannot get private storage from kernel

```
vxvm:vxconfigd: Error: Cannot get private storage from kernel
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.13 Cannot get private storage size from kernel

```
vxvm:vxconfigd: Error: Cannot get private storage size from
kernel
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.14 Cannot get record from the kernel

```
vxvm:vxconfigd: Error: Cannot get record name from the kernel:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.15 *Cannot make directory*

```
vxvm:vxconfigd: Error: Cannot make directory directory_path
```

♦ **Clarification**

When trying to create the specified directory, vxconfigd got a failure.

♦ **User Action**

Try creating the directory manually and then issue the command vxdctl enable.

A.1.1.16 *Cannot recover operation in progress*

```
vxvm:vxconfigd: Error: Cannot recover operation in progress
Failed to get group group_name from the kernel
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.17 Cannot start volume, no valid complete plexes

```
vxvm:vxconfigd: Error: Cannot start usage_type volume, no valid
complete plexes
```

♦ **Clarification**

No usable plexes remain for either the root or swap volume. This error is fatal and will result in the message `System startup failed` also appearing and the system being shutdown.

♦ **User Action**

This is generally an unrecoverable error and will likely require a reload of the system from backups.

A.1.1.18 Cannot start volume, no valid plexes

```
vxvm:vxconfigd: Error: Cannot start usage_type volume, no valid
plexes
```

♦ **Clarification**

No usable plexes remain for either the root or swap volume. This error is fatal and will result in the message `System startup failed` also appearing and the system being shutdown.

♦ **User Action**

The user can attempt to use the -f flag to force the operation. If successful, the administrator will need to take action to guarantee the consistency and correctness of the data. This is generally an unrecoverable error and will likely require a reload of the system from backups.

A.1.1.19 Cannot start volume, volume state is invalid

```
vxvm:vxconfigd: Error: Cannot start usage_type volume, volume
state is invalid
```

♦ **Clarification**

The volume is not in a state that can be recovered from. This might be because of corruption of the databases or because of an invalid use of the `vxconfigd` interfaces without the use of the utilities.

♦ **User Action**

The user can attempt to use the -f flag to force the operation. If successful, the administrator will need to take action to guarantee the consistency and correctness of the data. This is generally an unrecoverable error and will require reloading of the system from backups.

A.1.1.20 Cannot store private storage into the kernel

```
vxvm:vxconfigd: Error: Cannot store private storage into the
kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.21 Differing version of vxconfigd installed

```
vxvm:vxconfigd: Error: Differing version of vxconfigd installed
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.22 Disk, group, device not updated with new host ID

```
vxvm:vxconfigd: Error: Disk disk_name, group group_name,
device device_name: not updated with new host ID Error:
error_message
```

♦ **Clarification**

If the host ID for a system is changed using the `vxdctl init` command then all disks in all imported disk groups will need to have the host ID changed to the new ID. If the host ID for a disk cannot be changed, then this message will be displayed. Other problems might also exist for this disk.

♦ **User Action**

The contents of the disk should be evacuated elsewhere and the disk should be re-initialized.

A.1.1.23 Disk group, Disk: Cannot auto-import group:

```
vxvm:vxconfigd: Error: Disk group group_name, Disk disk_name:
Cannot auto-import group: error_message
```

♦ **Clarification**

The disk group *group_name*, could not be reimported after a system restart. The reason is given as part of the error message. Other error messages may appear which provide more information on what went wrong. Any volumes in the disk group will be unavailable until the error condition is fixed and the disk group is reimported.

♦ **User Action**

Clear the error condition, if possible, and then import the disk group by hand with `vxdg import`. After importing, you should restart all volumes with `vxdg -g` *`groupname`* `-sb`.

A.1.1.24 Disk group, Disk: Group name collides with record in rootdg

```
vxvm:vxconfigd: Error: Disk group group_name, Disk disk_name: Group
name collides with record in rootdg
```

♦ **Clarification**

The disk group name *group_name*, for the disk group being imported from the named disk, collides with a configuration record in the `rootdg` disk group. Disk groups must have names that do not match any records in the root disk group.

♦ **User Action**

If you wish to import the disk group, you will have to rename the conflicting record in `rootdg` to some other name.

A.1.1.25 Disk group: Cannot recover temp database

```
vxvm:vxconfigd: Error: Disk group group_name: Cannot recover temp
database:
error_message
```

♦ **Clarification**

The temp database stored in the root file system could not be opened or read. Other messages will detail the error. This may happen because of an I/O error or a problem in the file system.

♦ **User Action**

The system should be rebooted and the operation retried.

A.1.1.26 Disk group: Disabled by errors

```
vxvm:vxconfigd: Error: Disk group group_name: Disabled by errors
```

♦ **Clarification**

This message can appear if the last configuration database or last kernel log area for a disk group became disabled. This could have been due to an I/O error or some other condition. Other messages preceding this one are likely to highlight the root cause.

♦ **User Action**

Any remaining active volumes should be backed up. The disk group will have to be re-initialized and the disks re-added to the group in order to recover.

A.1.1.27 Disk group: Errors in some configuration copies

```
vxvm:vxconfigd: Error: Disk group group_name: Errors in some
configuration copies
```

♦ **Clarification**

One or more on-disk database copies were found to contain errors. As a result, the disk group could not be imported. This is most likely to be due to a disk I/O error, or to blocks of a configuration copy being overwritten within invalid contents. Check for messages from the disk driver. Errors pertaining to specific configuration copies are listed on successive lines. These lines can be in either of the following forms: File *filename*: *error_message*: Block *number*: *error_message* Disk *diskname*, copy *copy_number*: *error_message*: Block *number*: *error_message* Lines beginning with `File` indicate an error in the special configuration copy file used for storing non-persistent disk group information. Lines beginning with `Disk` indicate failure of a persistent configuration copy stored on a disk. The copy number indicates which of the disk's configuration copies contains the error.

♦ **User Action**

If one or more disks for the disk group are currently inaccessible (such as due to a cabling error), make the disks accessible and try to import the disk group again with `vxdg import`. Otherwise, the disk group is probably no longer usable and will have to be recreated. All volume configuration information for the disk group is lost.

A.1.1.28 Disk group: Reimport of disk group failed

```
vxvm:vxconfigd: Error: Disk group group_name: Reimport of disk
group failed: error_message
```

♦ **Clarification**

The reload of a disk group into the kernel failed. This could be because the log size for the kernel may not be set or because of some other error in the import procedure. Other messages should indicate the true cause of the failure.

♦ **User Action**

The operation should be retried unless some other error message leads to a suggested course of action. Failing this, the system should be rebooted.

A.1.1.29 *Disk group: update failed*

```
vxvm:vxconfigd: Error: Disk group group_name: update failed:
error_message
```

♦ **Clarification**

This message occurs because a database update failed completely. No complete copy of the database could be written for the disk group. The disk group will be disabled and further access for configuration changes will be disallowed. If this error occurs for the root disk group, it will probably be necessary to re-install the system.

♦ **User Action**

Any volumes still active in the disk group should be backed up. The disk group will then have to be re-initialized and the disks re-added to it.

A.1.1.30 *Exec of /sbin/vxiod failed*

```
vxvm:vxconfigd: Error: Exec of /sbin/vxiod failed
```

♦ **Clarification**

An `exec` of `/sbin/vxiod` failed.

♦ **User Action**

Check the existence and permissions of the `/sbin/vxiod` command. Try executing the command manually to ensure that it can be run.

A.1.1.31 Failed to store commit status list into kernel

```
vxvm:vxconfigd: Error: Failed to store commit status list into
kernel: error_message
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.32 Fork of logio daemon failed

```
vxvm:vxconfigd: Error: Fork of logio daemon failed
```

♦ **Clarification**

The creation of a process that could then be used as a logging daemon failed.

♦ **User Action**

Check for messages explaining the reason that a `fork`(2) call failed. Retry the operation.

A.1.1.33 *GET_VXINFO ioctl failed, Version number of kernel does not match vxconfigd*

```
vxvm:vxconfigd: Error: GET_VXINFO ioctl failed:
error_message
vxvm:vxconfigd: Error: Version number of kernel does not match
vxconfigd
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.34 *Get of current rootdg failed*

```
vxvm:vxconfigd: Error: Get of current rootdg failed
```

♦ **Clarification**

An attempt to retrieve the rootdg from the kernel failed. This might be because of a kernel `vxconfigd` inconsistency or could also be because of a version difference between `vxconfigd` and the kernel.

♦ **User Action**

Check that the correct version of `vxconfigd` and the kernel are installed. Other messages might suggest other problems in a prior attempt at loading a configuration and possible courses of action. Failing that, contact Customer Support.

A.1.1.35 *No convergence between root disk group and disk list*

```
vxvm:vxconfigd: Error: No convergence between root disk group and
disk list
Disks in one version of rootdg:
disk_name type=disk_type info=disk_info
Disks in alternate version of rootdg:
disk_name type=disk_type info=disk_info
```

♦ **Clarification**

This message can appear when `vxconfigd` is not running in autoconfigure mode (see the `vxconfigd`(1m) manual page) and when, after several retries, it can not resolve the set of disks belonging to the root disk group. The algorithm for non-autoconfigure disks is to scan disks listed in the `/etc/vx/vxboot` file and then examine the disks to find a database copy for the rootdg disk group. The database copy is then read to find the list of disk access records for disks contained in the group. These disks are then examined to ensure that they contain the same database copy. As such, this algorithm expects to gain convergence on the set of disks and the database copies contained on them. If a loop is entered and convergence cannot be reached, then this message will appear and the root disk group importation will fail.

♦ **User Action**

Reorganizing the physical locations of the devices attached to the system may break the deadlock. Failing this contact Customer Support.

A.1.1.36 *Open of directory failed*

```
vxvm:vxconfigd: Error: Open of directory directory_path failed
```

♦ **Clarification**

When `vxconfigd` was trying to create node files for the volumes, it was unable to open the directory in which the nodes were to be created.

♦ **User Action**

Check for other errors that might suggest why the directory might be missing or if the permissions might be incorrect. Fix the condition to allow vxconfigd to open or create the directory, then issue the command `vxdctl enable`.

A.1.1.37 *Read of directory failed*

```
vxvm:vxconfigd: Error: Read of directory directory_path failed
```

♦ **Clarification**

The node directory could not be read when `vxconfigd` was trying to scan for volume nodes.

♦ **User Action**

Check for other messages that might suggest why the directory is inaccessible. Try reading the directory manually if the directory is corrupted, then try removing and re-creating it and then restarting `vxconfigd`.

A.1.1.38 *System boot disk does not have a valid plex*

```
vxvm:vxconfigd: Error: System boot disk does not have a valid
usage_type plex
Please boot from one of the following disks:

Disk: disk_media_name    Device: disk_access_name
```

♦ **Clarification**

If the system was booted on a drive other than a drive containing an active plex of the root file system, then this message will appear. Since the unix file and init processing has already occurred from that drive and since root has been mounted read-only, it is not reasonable for VxVM to restore the contents of the stale device since that may have the effect of changing some already-read data. Instead, the best that can be done is to recommend the correct device to boot from.

♦ **User Action**

Follow the instructions for rebooting from an alternate drive and select one of the listed drives.

A.1.1.39 System startup failed

```
vxvm:vxconfigd: Error: System startup failed
```

♦ **Clarification**

Some part of the root volume start procedure failed. Other messages should indicate the reason. The system will be shutdown and left in a halted state.

♦ **User Action**

This is generally an unrecoverable error and will require either the boot of an alternate root device, or reloading of the system from backups. The messages proceeding this one may suggest some other course of action.

A.1.1.40 There is no volume configured for the device

```
vxvm:vxconfigd: Error: There is no volume configured for the
usage_type device
```

♦ **Clarification**

The configuration for the root or swap volume is not complete, such that there is no configuration known to vxconfigd with the required minor number or else the usage-type field in the volume structure does not match either the root or swap usage type required. This error is fatal and will result in the message System startup failed also appearing and the system being shutdown.

♦ **User Action**

This is generally an unrecoverable error and will require either the boot of an alternate root device, or reloading of the system from backups.

A.1.1.41 Unexpected configuration tid for group found in kernel

```
vxvm:vxconfigd: Error: Unexpected configuration tid for group
group_name found in kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.42 Unexpected error during volume reconfiguration

```
vxvm:vxconfigd: Error: Unexpected error during usage_type volume
reconfiguration:
error_message
```

♦ **Clarification**

A record lock for the volume could not be acquired as part of the initial volume setup for either a `root` or `swap` volume. This is most likely to occur under low memory conditions.

♦ **User Action**

Other messages may suggest an alternate course of action. Otherwise, this is generally an unrecoverable error and will require either the boot of an alternate `root` device or reloading of the system from backups.

A.1.1.43 Unexpected error fetching disk for volume

```
vxvm:vxconfigd: Error: Unexpected error fetching disk for
usage_type volume:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.44 Unexpected values stored in the kernel

```
vxvm:vxconfigd: Error: Unexpected values stored in the kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.45 *VOL_RESET_KERNEL failed: a volume or plex device is open*

```
vxvm:vxconfigd: Error: VOL_RESET_KERNEL failed: a volume or plex
device is open
or
vxvm:vxconfigd: Error: VOL_RESET_KERNEL failed: error_message
```

♦ **Clarification**

An attempt at resetting the kernel state with a `vxconfigd -r reset` command failed because all the Volume Manager objects in the kernel were not closed. If any volumes are in use, then the reset cannot be performed. This may also happen if a reset was requested on a system with root volumes. Root volumes are, by definition, never closed and so a reset cannot be performed.

♦ **User Action**

If a reset is really desired, then checking the state of the volumes and any mounted file systems should result in information about who might have them open. Unmounting all volumes and killing any processes accessing the volumes should allow the reset to occur.

A.1.1.46 *Unrecognized operating mode*

```
vxvm:vxconfigd: Error: mode: Unrecognized operating mode
```

♦ **Clarification**

An unknown mode string was entered following a `-m` option.

♦ **User Action**

Select a valid mode from the `vxconfigd`(1m) manual page and try again.

A.1.1.47 *cannot open /dev/vx/iod*

```
vxvm:vxconfigd: Error: cannot open /dev/vx/iod: error_message
```

♦ **Clarification**

The open of the `/dev/vx/iod` file can only fail if the device node is missing or has an incorrect major or minor number.

♦ **User Action**

Check the existence and values of the file and make sure that VxVM was correctly installed.

A.1.1.48 *cannot open argument*

```
vxvm:vxconfigd: Error: cannot open argument: error_message
```

♦ **Clarification**

The tracefile specified on the command line could not be opened in append mode. The error message supplied should explain the reason.

♦ **User Action**

Select an alternate tracefile name that can be created or appended to.

A.1.1.49 *cannot open vxconfig_device*: Device is already open

```
vxvm:vxconfigd: Error: cannot open vxconfig_device: Device is
already open
or
vxvm:vxconfigd: Error: cannot open vxconfig_device:
error_message
```

♦ **Clarification**

The exclusive open device (/dev/vx/config) is already open. Only one vxconfigd process can be active on the system at one time. Subsequent attempts at starting vxconfigd or opening the device will result in this message.

♦ **User Action**

Check for other running vxconfigd processes. vxdctl mode will report if vxconfigd is currently active. Otherwise a "fuser(1m)" command may give clues as to who has the device open.

A.1.1.50 *enable failed*

```
vxvm:vxconfigd: Error: enable failed: error_message
```

♦ **Clarification**

This message may occur during an initial startup of vxconfigd. If changing to enabled mode when this error occurs, failures could be due to problems with the creation of the portal or with connection to the kernel. If changing from an enabled state to a disabled state, then problems could occur with removing the disk groups from the kernel because of such things as volumes in use.

♦ **User Action**

Evaluate other error messages occurring with this one to determine the root cause of the problem. Make changes suggested by the other errors and then retry the command.

A.1.1.51 *failed to create daemon: fork failed*

```
vxvm:vxconfigd: Error: failed to create daemon: fork failed:
error_message
```

♦ **Clarification**

The call to fork(2) to generate a background vxconfigd process failed.

♦ **User Action**

Check for messages explaining the reason that a `fork`(2) call failed. Retry the operation.

A.1.1.52 *Wait for logging daemon failed*

```
vxvm:vxconfigd: Error: volume volume_name: Wait for logging
daemon failed
```

♦ **Clarification**

The wait called to wait for the existence of the daemon process did not execute correctly. This can only happen if the ioctl does not correctly match the command required, perhaps because of a mismatch between the `vxiod` command and the kernel versions or perhaps because of an incorrect minor number for the `/dev/vx/iod` device.

♦ **User Action**

Check the existence and permissions of the `/dev/vx/iod` device.

A.1.1.53 *Disk group rootdg: Inconsistency -- Not loaded into kernel*

```
vxvm:vxconfigd: FATAL Error: Disk group rootdg: Inconsistency --
Not loaded into kernel
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.54 Cannot update kernel

```
vxvm:vxconfigd: FATAL Error: Group group_name: Cannot update
kernel
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.55 Interprocess communication failure

```
vxvm:vxconfigd: FATAL Error: Interprocess communication failure:
error_message
```

♦ **Clarification**

The portal to client utilities has returned a failure. This is a fatal error since without a portal to clients, vxconfigd cannot do anything useful.

♦ **User Action**

Check for other errors suggesting the reason for portal failure. Restart vxconfigd. If problems persist, reboot the system.

A.1.1.56 Invalid status stored in kernel

```
vxvm:vxconfigd: FATAL Error: Invalid status stored in kernel
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.57 Memory allocation failure during startup

```
vxvm:vxconfigd: FATAL Error: Memory allocation failure during
startup
```

♦ **Clarification**

vxconfigd could not allocate sufficient memory space to perform some operation.

♦ **User Action**

Restart vxconfigd. You may need to kill off some running processes or add more swap space.

A.1.1.58 Rootdg cannot be imported during boot

```
vxvm:vxconfigd: FATAL Error: Rootdg cannot be imported during
boot
```

♦ **Clarification**

If one of the preceding messages was produced and was an unrecoverable error, then this message will be displayed. This is a fatal error for the boot process.

♦ **User Action**

Booting from an alternate root device should be attempted. If this fails, then the root disk will have to be restored from an alternate media, or the system will have to be reloaded. If the alternate device reboot succeeds, then the source reason for the failure as described in the prior error messages will then need to be fixed before continuing with use of the system.

A.1.1.59 *vxconfigd_SDI_INFO ioctl failed*

```
vxvm:vxconfigd: FATAL Error: vxconfigd_SDI_INFO ioctl failed:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.60 *Cannot change disk group record in kernel*

```
vxvm:vxconfigd: Warning: Cannot change disk group record in
kernel: error_message
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.61 Cannot create device path

```
vxvm:vxconfigd: Warning: Cannot create device path: error_message
```

♦ **Clarification**

The mknod(2) call made by vxconfigd to create a device node failed. The reason for the error should be displayed.

♦ **User Action**

Fix the reason indicated for node creation failure and then issue the command vxdctl enable.

A.1.1.62 Cannot exec /bin/rm to remove directory_path

```
vxvm:vxconfigd: Warning: Cannot exec /bin/rm to remove
directory_path: error_message
```

♦ **Clarification**

An exec of /sbin/rm failed.

♦ **User Action**

Ignore the error. It is not serious if the directory could not be removed.

A.1.1.63 Cannot fork to remove directory directory_path

```
vxvm:vxconfigd: Warning: Cannot fork to remove directory
directory_path: error_message
```

♦ **Clarification**

The call to fork(2) to generate a process failed.

♦ **User Action**

Ignore the error. It is not serious if the directory could not be removed.

A.1.1.64 *Disk device_name* in kernel not a recognized type

```
vxvm:vxconfigd: Warning: Disk device_name in kernel not a
recognized type
```

♦ **Clarification**

The disk type of a disk in the kernel does not match any known disk type. This can only occur if vxconfigd and the kernel are in an inconsistent state.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a deinstallation and reinstallation of the VxVM package could be attempted. Failing this, contact Customer Support.

A.1.1.65 Disk *disk_name* names group *group_name*, but group ID differs

```
vxvm:vxconfigd: Warning: Disk disk_name names group group_name,
but group ID differs
```

♦ **Clarification**

As part of a disk group import, a disk was discovered that had a mismatched disk group name and disk group ID. This disk will not have been imported. This can only happen if two disk groups of the same name exist that have different disk group ID values. In that case, one group will be imported along with all its disks and the other group will not. This message will appear for disks in the un-selected group.

♦ **User Action**

If it turns out that the disk should be imported into the group, then this will have to be done by adding the disk to the group at a later stage. It will not happen automatically as part of the import. All configuration information for the disk will also be lost.

A.1.1.66 Disk group *group_name* is disabled, disks not updated with new host ID

```
vxvm:vxconfigd: Warning: Disk group group_name is disabled, disks
not updated with new host ID
```

♦ **Clarification**

If the host ID for a system is changed using the `vxdctl init` command then all disks in all imported disk groups will need to have the host ID changed to the new ID. If a disk group is found in the imported but disabled state, then the host ID will not be changed.

♦ **User Action**

The host ID will need to be cleared using the `vxdisk clearimport` command for each disk, and then the disk group should be re-imported.

A.1.1.67 *Disk group log may be too small*

```
vxvm:vxconfigd: Warning: Disk group group_name: Disk group log
may be too small. Log size should be at least number blocks
```

♦ **Clarification**

The log areas for the disk group have become too small for the size of configuration currently in the group. This should normally never happen without first displaying a message about the database area size. This message only occurs during disk group import; it can only occur if the disk was inaccessible while new database objects were added to the configuration, and the disk was then made accessible and the system restarted.

♦ **User Action**

If this situation does occur, then the disks in the group will have to be explicitly re-initialized with larger log areas. See the manual page for `vxdisk`(1m). To reinitialize all the disks, they must be detached from the group with which they are associated and then reinitialized and readded. The group should then be deported and re-imported for the changes to the log areas for the group to take effect.

A.1.1.68 Errors in some configuration copies

```
vxvm:vxconfigd: Warning: Disk group group_name: Errors in some
configuration copies:
```

♦ **Clarification**

One or more on-disk database copies were found to contain errors. As a result, the disk group could not be imported. This is most likely to be due to a disk I/O error, or to blocks of a configuration copy being overwritten within invalid contents. Check for messages from the disk driver. Providing that other copies of the database can be successfully read, the system will continue and the disk group import or initial `vxconfigd` enable operation should succeed. If the database copy can subsequently be written to, then this message will not recur.

Errors pertaining to specific configuration copies are listed on successive lines. These lines can be in either of the following forms: File *filename*: *error_message*: Block *number*: *error_message* Disk *diskname*, copy *copy_number*: *error_message*: Block *number*: *error_message* Lines beginning with `File` indicate an error in the special configuration copy file used for storing non-persistent disk group information. Lines beginning with `Disk` indicate failure of a persistent configuration copy stored on a disk. The copy number indicates which of the disk's configuration copies contains the error.

♦ **User Action**

This message is likely to occur once due to an I/O failure and then not recur. If it does recur, then it may be necessary to remove the disk and re-initialize it to clear the condition. If all configuration copies for a disk group become unusable, then the disk group itself becomes unusable and must be recreated. If the `rootdg` disk group becomes unusable, the Volume Manager may need to be deinstalled and reinstalled. In this case, if root file system is on a volume, then the operating system itself may need to be reinstalled.

A.1.1.69 *Error in vxboot file*

```
vxvm:vxconfigd: Warning: Error in vxboot file: error_message
Entry: disk disk_name disk_type disk_info
```

♦ **Clarification**

This message occurs when an entry in the `vxboot` file does not contain the correct information to define a valid disk access record.

♦ **User Action**

The entry should probably be removed using the `vxdctl rmdisk` command and then re-added using `vxdctl adddisk`.

A.1.1.70 `Failed to update vxconfigd info area in kernel`

```
vxvm:vxconfigd: Warning: Failed to update vvxconfigd info area in
kernel:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.71 *Field too long in vxboot file*

```
vxvm:vxconfigd: Warning: Field too long in vxboot file:
Entry: disk disk_name disk_type disk_info
```

♦ **Clarification**

The vxboot file is maintained by vxconfigd and vxdctl and should never normally exhibit this problem. This problem might indicate some corruption of the vxboot file or could also be the result of manual editing of the file.

♦ **User Action**

The offending entry could try to be removed by use of the vxdctl rmdisk command. Failing that, vxboot may have to be re-initialized using a vxdctl init command.

A.1.1.72 Get of record *record_name* from kernel failed

```
vxvm:vxconfigd: Warning: Get of record record_name from kernel
failed:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between vxconfigd and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of vxconfigd or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting vxconfigd. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.73 Plex for volume is stale or unusable

```
vxvm:vxconfigd: Warning: Plex plex_name for usage_type volume is
stale or unusable
```

♦ **Clarification**

This message is output to alert the user to the failure of one or more plexes of either the root or swap volume. The system may be able to continue depending on the existence of other usable plexes for the volume.

♦ **User Action**

The failed plex should be repaired by either re-attaching the plex to the volume once the system is booted, or else by evacuating and replacing the disk on which the failed plex resides if it is thought that the disk is going bad.

A.1.1.74 *cannot remove group from kernel*

```
vxvm:vxconfigd: Warning: cannot remove group group_id from
kernel:
error_message
```

♦ **Clarification**

Some inconsistency has arisen between `vxconfigd` and the kernel and has caused an ioctl to fail. This could be caused by the use of older versions of `vxconfigd` or the kernel, or it could be due to a bug in the Volume Manager.

♦ **User Action**

Try stopping and restarting `vxconfigd`. If this fails then a reboot of the system should be attempted, possibly followed by a deinstallation and reinstallation of the VxVM package. Failing this, contact Customer Support.

A.1.1.75 *response to client failed*

```
vxvm:vxconfigd: Warning: response to client client_number failed:
error_message
```

♦ **Clarification**

The portal to client utilities has returned a failure. This is a fatal error since without a portal to clients, `vxconfigd` cannot do anything useful. This could be caused by a STREAMS error or some other communications problem with the client.

♦ **User Action**

Check for other errors suggesting the reason for portal failure. Restart `vxconfigd`. If problems persist, reboot the system.

A.1.2 Kernel Error Messages

The following are the kernel level error messages.

Recovery

B.1 Introduction

Disk failures can cause two types of problems: loss of data on the failed disk and loss of access to your system due to the failures of a key disk (a disk involved with system operation). The VERITAS Volume Manager provides the ability to protect your system from either type of problem. VxVM allows you to use *mirroring* to protect the data. By mirroring your data, you prevent data loss from a disk failure. By mirroring drives critical to system operation, you ensure that no single disk failure will leave your system unusable.

B.2 Plex and Volume States

The following sections describe the plex and volume states.

B.2.1 Plex States

Plex states reflect whether or not plexes are complete and consistent copies (mirrors) of the volume contents. VxVM utilities automatically maintain the plex state. However, a system administrator can modify the state of a plex if changes to the volume with which the plex is associated should not be written to it. For example, if a disk with a particular plex located on it begins to fail, that plex can be temporarily disabled.

Note – A plex does not have to be associated with a volume. A plex can be created with the command line interface.

The plex created with this command can later be attached to a volume if required.

VxVM utilities use plex states to:

- indicate whether volume contents have been initialized to a known state
- determine if a plex contains a valid copy (mirror) of the volume contents
- track whether a plex was in active use at the time of a system failure
- monitor operations on plexes

This section explains plex states in detail. This section is designed for users who wish to have a detailed knowledge of plex states.

Plexes that are associated with a volume always have one of the following states:

- EMPTY
- CLEAN
- ACTIVE
- STALE
- OFFLINE
- TEMP
- TEMPRM
- IOFAIL

B.2.1.1 EMPTY Plex State

Volume creation sets all plexes associated with the volume to the EMPTY state to indicate to the usage type utilities.

B.2.1.2 CLEAN Plex State

A plex is in a CLEAN state when it is known to contain a consistent copy (mirror) of the volume contents and a operation has disabled the volume. As a result, when all plexes of a volume are clean, no action is required to guarantee that the plexes are identical when that volume is started.

B.2.1.3 ACTIVE Plex State

A plex can be in the ACTIVE state in two situations:

- when the volume is started and the plex fully participates in normal volume I/O (meaning that the plex contents change as the contents of the volume change)
- when the volume was stopped as a result of a system crash and the plex was ACTIVE at the moment of the crash

In the latter case, a system failure may leave plex contents in an inconsistent state. When a volume is started, VxVM performs a recovery action to guarantee that the contents of the plexes that are marked as ACTIVE are made identical.

Note – On a well running system, ACTIVE should be the most common state you see for any volume's plexes.

B.2.1.4 STALE Plex State

If there is a possibility that a plex does not have the complete and current volume contents, that plex is placed in the STALE state. Also, if an I/O error occurs on a plex, the kernel stops using and updating the contents of that plex, and a operation sets the state of the plex to STALE.

A `vxplex attach` operation revives STALE plexes from an ACTIVE plex. Atomic copy operations copy the contents of the volume to the STALE plexes. The system administrator can force a plex to the STALE state with an utility operation.

B.2.1.5 OFFLINE Plex State

The `vxmend off` operation indefinitely detaches a plex from a volume by setting the plex state to OFFLINE. Although the detached plex maintains its association with the volume, changes to the volume do not update the OFFLINE plex until the plex is put online and reattached with the `vxplex att` operation. When this occurs, the plex is placed in the STALE state, which causes its contents to be recovered at the next `volume start` operation.

B.2.1.6 TEMP Plex State

Setting a plex to the TEMP state facilitates some plex operations that cannot occur in a truly atomic fashion. For example, attaching a plex to an enabled volume requires copying volume contents to the plex before it can be considered fully attached.

A utility will set the plex state to TEMP at the start of such an operation and to an appropriate state at the end of the operation. If the system goes down for any reason, a TEMP plex state indicates that the operation is incomplete; a subsequent `volume start` will dissociate plexes in the TEMP state.

B.2.1.7 TEMPRM Plex State

A TEMPRM plex state resembles a TEMP state except that at the completion of the operation, TEMPRM plex is removed. Some subdisk operations require a temporary plex. Associating a subdisk with a plex, for example, requires updating the subdisk with the volume contents before actually associating the subdisk. This update requires associating the subdisk with a temporary plex, marked TEMPRM, until the operation completes and removes the TEMPRM plex.

If the system goes down for any reason, the TEMPRM state indicates that the operation did not complete successfully. A subsequent operation will dissociate and remove TEMPRM plexes.

B.2.1.8 IOFAIL Plex State

The IOFAIL plex state is associated with persistent state logging. On the detection of a failure of an ACTIVE plex, `vxconfigd` places that plex in the IOFAIL state so that it is disqualified from the recovery selection process at volume start time.

B.2.2 The Plex State Cycle

The changing of plex states accompanies normal operations. Deviations in plex state indicate abnormalities that VxVM must normalize. At startup, the `volume start` operation makes all CLEAN plexes ACTIVE. If all goes well until shutdown, the volume-stopping operation marks all ACTIVE plexes CLEAN and the cycle continues. Having all plexes CLEAN at startup (before `volume start` makes them ACTIVE) indicates a normal shutdown and optimizes startup.

If a crash occurred, the volume-starting operation finds no CLEAN plexes, only ACTIVE ones. The operation then establishes one plex as an up-to-date and suitable source for reviving the other plexes, and marks that source plex ACTIVE and the others STALE. The volume usage type determines which plex is selected as the source plex.

If an I/O error occurred and caused a plex to become disabled, the volume-stopping operation marks the plex in which the error occurred as STALE. Any STALE plexes require recovery. When the system restarts, a utility copies data from an ACTIVE to a STALE plex and makes the STALE plex ACTIVE.

B.2.3 Plex Kernel State

The *plex kernel state* indicates the accessibility of the plex. The plex kernel state is monitored in the volume driver and allows a plex to have an off-line (DISABLED), maintenance (DETACHED), and on-line (ENABLED) mode of operation.

- DISABLED — The plex may not be accessed.
- DETACHED — A write to the volume is not reflected to the plex. A read request from the volume will never be satisfied from the plex device. Plex operations and ioctl functions are accepted.

- ENABLED — A write request to the volume will be reflected to the plex, if the plex is set to ENABLED for write mode. A read request from the volume is satisfied from the plex if the plex is set to ENABLED.

Note – No user intervention is required to set these states, they are maintained internally. On a system that is operating properly, all mirrors are enabled.

B.2.4 Volume States

There are four volume states, some of which are similar to plex states:

- CLEAN—The volume is not started (kstate is DISABLED) and it's plexes are synchronized.
- ACTIVE—The volume has been started (kstate is currently ENABLED) or was in use (kstate was ENABLED) when the machine was rebooted. If the volume is currently ENABLED, the state of its plexes at any moment is not certain (since the volume is in use). If the volume is currently DISABLED, this means that the plexes cannot be guaranteed to be consistent.
- EMPTY—The volume contents are not initialized. The kernel state (kstate) is always DISABLED when the volume is EMPTY.
- SYNC—The volume is either in read-writeback mode (kstate is currently ENABLED) or was in read-writeback mode when the machine was rebooted (kstate is DISABLED). If the volume is ENABLED, this means that the plexes are being resynchronized via the read-writeback recovery. If the volume is DISABLED, it means that the plexes were being resynchronized via read-writeback when the machine rebooted and therefore still need to be synchronized.

The interpretation of these flags during volume startup is modified by the persistent state log for the volume (for example, the dirty/clean flag). If the clean flag is set, this means that an ACTIVE volume was not written to by any processes or was not even open at the time of the reboot; therefore, it can be considered CLEAN. The clean flag will always be set in any case where the volume is marked CLEAN.

B.2.5 Volume Kernel State

The *volume kernel state* indicates the accessibility of the volume. The volume kernel state allows a volume to have an off-line (DISABLED), maintenance (DETACHED), and on-line (ENABLED) mode of operation.

- DISABLED—The volume cannot be accessed.
- DETACHED—The volume cannot be read or written, but plex device operations and ioctl functions are accepted.
- ENABLED—The volumes can be read and written.

B.3 Protecting Your System

In order to maintain system availability, the data important to running and booting your system must be mirrored. Furthermore, it must be preserved in such a way that it can be used in case of failure. The most difficult part of this is the ability to boot the system after a failure of a disk that is critical to the boot process.

To preserve data, create and use volumes that have at least two mirrors (plexes). The mirrors must be on different disks. The `vxassist` utility locates the mirrors such that the loss of one disk will not result in a loss of data. Edit the file `/etc/default/volassist` to set the default number of mirrors for newly created volumes to two.

You must also do regular backups (of all data except the root file system) to protect your data. Backups are necessary if all copies of a volume are lost or corrupted in some way. For example, a power surge could damage several (or all) disks on your system. Alternately, a mistyped command could remove critical files or damage a file system directly.

In such cases, boot the system from the CDROM and restore the `/etc/vfstab`.

Also forceload all the drivers required for the root mirror disks. The driver names for these disks could be found from the name of these disks in the `/devices` directory.

If the system file is damaged or lost, and a backup copy of the system file is not available, and /usr is a volume, the system must be booted from the CDROM. Mount one of the root partitions and edit the system file on it. Enter the following lines in the system file:

```
* vxvm_START
rootdev:/pseudo/vxio@0:0
set vxio:vol_rootdev_is_volume=1
set vxio:vol_swapdev_is_volume=1
set vxio:usrport_is_volume=1
* vxvm_END
```

Also forceload all the drivers required for the root mirror disks. After these changes, reboot the system from the same root partition on which the system file was restored.

B.4 The UNIX Boot Process

The boot process for UNIX depends on the ability for the system hardware to find programs and data that are necessary to bring the system up. The system boots in several stages, each successive stage depending on the previous stage. Different combinations of system hardware and disk controllers provide different capabilities for configuring your system which impact the boot process. Before deciding how to configure your system, it is important to have some understanding of the boot process and how different controllers affect it.

The boot process starts when the system is turned on or reset. The first thing that is run is the Basic Input/Output System (BIOS) initialization routine or some other ROM-based boot code. This routine serves several purposes: it performs some simple diagnostics (like checking memory); it loads default values for the system configuration, such as the disk configuration; and it scans the system bus for any peripherals attached to the system.

One system administrator concern is the configuration of the disks. Intel 386 systems typically conform to the IBM PC BIOS interface, and initially configure four disks for the system: two floppy drives, usually noted as `A:` and `B:`, and two hard drives, `C:` and `D:`. This configuration is normally kept in non-volatile RAM (NVRAM) on the system, and is configurable through a system-specific interface (some BIOSs have on-board configuration capabilities; others require

a system floppy to reconfigure the system). The four disk devices are the disks that are available for use through the BIOS interface, and in most cases, these are the only devices available during the early stages of the boot process, until UNIX is actually loaded and running. (Some disk controllers allow access to more than two hard drives during the early stages of booting; more on this later.)

Once the default configuration is loaded and checked by the system BIOS, the BIOS initialization routines will check to see if any peripherals on the bus have their own BIOS and initialization routines and, if so, it will run them. This allows attached hardware to initialize itself and change the default configuration if they deem it necessary. Many disk controllers have their own BIOS routines and will change the default configuration so that the `C:` and `D:` drive entries in the system configuration point to drives that are attached to that controller.

Once all peripheral BIOS routines have run, the system will attempt to boot an operating system from one of the disk devices. It first checks the `A:` floppy drive to see if a floppy is inserted. If the drive `A:` does not contain a floppy, the BIOS attempts to read and execute a program, called the `fdisk boot`, from the first block of the drive designated as `C:`. This program then goes on to read and execute another program from the disk's active partition (which is the UNIX partition), called the UNIX `boot` program. This program then does some preparation for the loading of the UNIX operating system, loads UNIX from the `/stand` file system on the disk, and starts up UNIX. The `boot` program also passes UNIX some information about the system configuration that will be used during the UNIX part of the boot process.

When UNIX is started, it examines the system and the arguments passed in from `boot` and does it's own initialization. It eventually mounts the root file system, sets up the initial swap area, and executes the `init` program (located in `/sbin/init`) to bring the system up. `init` runs the VxVM startup routines which load the complete Volume Manager configuration, check the root file system, etc.

B.4.1 Disk Controller Specifics

As mentioned above, disk controllers are given the opportunity to change the system configuration for the `C:` and `D:` disk locations during the BIOS initialization process. The exact actions taken depend entirely on the controller involved. Some controllers are very simple and just map the first two disks it

finds into C: and D:; more advanced controllers are capable of being configured to use specific devices or to check for failed disks and, if possible, substitute others. The basic function, however, is to point the entry for disk C: in the system BIOS configuration at the disk that should be used for the rest of the boot process (e.g., where to find fdisk boot, the UNIX boot, and the UNIX operating system itself). If no disk is configured as C:, or if that disk does not have the necessary contents for booting UNIX, the boot will fail.

B.4.1.1 SCSI Controllers

While the specifics of what any controller will do is entirely dependent on the type of the controller, there are some general capabilities that a controller can have that can help the boot process. The basic SCSI controller will map the disk with a SCSI ID of 0 into C: and the disk with a SCSI ID of 1 into D:. Other controllers give the administrator other configuration options and features. While these vary greatly between controllers, it is possible to classify controller features into three groups. These groups are:

- Simple controllers are exactly that - simple. They will only map SCSI disk 0 into C: and SCSI disk 1 into D:. Some simple controllers (such as the Adaptec 1540/1542 B) will not map SCSI disk 1 into D: if no SCSI disk 0 responds on the bus. This can be inconvenient, since a failure of SCSI disk 0 will require that the administrator reconfigure the disks on the controller in order to boot the system.
- These controllers allow the administrator to specify a configuration for mapping the disks attached to C: and D:.
- Auto-failover controllers give the maximum convenience to the administrator. When mapping disks to C: and D:, it will do some kind of validation of those disks (such as making sure the specified disk is still attached to the controller and is powered on). If a specified disk fails the validation, another disk on the controller is chosen and mapped into the configuration at the location of the failed disk.

Note that some controllers mix the above characteristics. For example, some non-configurable controllers will perform auto-failover. This is usually done by having the controller poll the SCSI bus and map the disk with the lowest SCSI ID into C: and the next lowest SCSI ID into D:. It should be noted that some auto-failover controllers can, at the time of installation, be configured to perform in the simple controller mode. It is essential that the user has full knowledge of the capabilities of the controller. Controller specific information

may be found in most user manuals available from the manufacturer. The mapping policy of a Micro-channel MCA SCSI controller is quite different from the policies explained above. Please see the section “Configuring the system” for Micro-channel controller specific information. No matter what kind of controller you have, it will attempt to map a disk into `C:`. This disk is usually referred to as the `boot` disk, since this is the disk that will be used for the early stages of the boot process. The system will not boot if:

- no disk is mapped into `C:` for the system BIOS to find
- the disk does not have the proper data on it for booting (such as the `fdisk boot` program)
- the boot data is corrupted (such as a corrupted fdisk partition on the disk).

For simple controllers, a bootable disk is normally placed on the controller in such a way that it will be mapped into `C:` by the controller. (For example, for the Adaptec 1540/1542 B controller, the administrator would be forced to replace the failed SCSI disk 0 with a properly configured disk and change its SCSI id to 0.) By mirroring the system’s boot-critical data to another disk with VxVM, that backup disk can be mapped into `C:` in case of a primary boot disk failure and can be used to bring up the system.

Unfortunately, rearranging the disks so that the backup boot disk is mapped into `C:` can mean disconnecting and reconnecting drives, moving jumpers to reorder the disks, etc., which is inconvenient. With some disk controllers, other disks besides `C:` are available for use during the boot process. Even with auto-failover controllers, the system may be unbootable because of a failure later in the boot process (such as an invalid UNIX partition table) that the controller cannot detect.

To avoid having rearrange the hardware, VxVM supports a special boot floppy available that, in many cases, can make the system boot from an alternate drive without having to rearrange hardware.

B.4.1.2 The VxVM Boot Floppy

The VxVM boot floppy gives the administrator the opportunity to designate a disk other than the one mapped to `C:` for use as the boot disk. This can be very convenient when the disk that is mapped into `C:` has failed completely or contains stale or corrupt data.

B.4.1.3 Booting With the VxVM Boot Floppy

When the VxVM boot floppy is inserted in floppy drive `A:`, the system BIOS will read the `fdisk boot` and `boot` programs from the floppy, circumventing data problems on the disk mapped into `C:`. The `boot` program on this floppy is slightly different than the normal hard-drive `boot` program. Once the system has initialized and the floppy `boot` program is running, the following prompt will appear on the screen:

```
Enter kernel name [C:unix]:
```

The kernel name specified can have two parts: a disk name and a file name. The default (as shown) is to boot unix from the disk mapped into `C:` (eg., `C:unix`). The administrator can now select an alternate by specifying the disk and/or the kernel by entering the disk identifier and/or the operating system name.

Note – In almost all cases, the kernel name should be unix; booting a different kernel can have negative effects on your system.

Since the operating system name will usually be unix, the administrator can simply enter the disk to be used as the boot disk. Entering `D:` and pressing Return will use the drive mapped into `D:` (if any exists) as the boot disk.

B.4.1.4 Creating a VxVM Boot Floppy

The `vxmkboot` utility is used to create VxVM boot floppies. To do this, place a formatted floppy in the first floppy drive on the system. (See the man pages for formatting a floppy and floppy devices.) The boot image is placed on the floppy issuing the command

```
/etc/vx/bin/vxmkboot
```

at a shell prompt. If successful, the command will output

xx`+0 records in`

xx`+0 records out`

where *xx* is a number indicating the size of the boot program. If a failure occurs, an error message describing the error will be printed and a message will be output indicating the VxVM boot floppy creation failed.

It is suggested that the administrator create several boot floppies after VxVM is installed and that these be kept in a safe place.

Note – The program for creating the boot floppies exists on the system. If the system fails no floppies can be created. Unless a boot floppy has already been made, none will be available to help the administrator boot and recover the system.

B.4.2 Configuring The System

The best way to configure your system to remain available when boot-critical data or disks are lost is dependent on the characteristics of the disk controller on your system. The basic idea is to mirror your boot disk to another disk that will be available during the early boot process in case the normal boot disk becomes unavailable. If the system is using an auto-failover disk controller, the system should be configured such that the controller will choose the backup boot disk to map to `C:` if the controller decides the normal boot disk has failed; this means the system will automatically use a backup disk and therefore reboot without manual intervention in the case of boot disk failures that are detected by the controller.

This section suggests system configurations for the three types of controllers described above.

B.4.2.1 Simple Controllers

Using simple controllers it is impossible to avoid hardware reconfiguration in case of a failed disk with SCSI id 0.

If disk 0 does not respond on the bus, the controller will not map any disk into `C:` or `D:`, making the system totally unbootable, even with the VxVM boot floppy. It always attempts to map SCSI disk 0 into `C:` and SCSI disk 1 into `D:`, and these are the only disks available during the system boot. Therefore, the best possible configuration is to use `vxrootmir` to mirror the boot disk to the disk with SCSI ID 1. This allows the administrator to boot using the VxVM boot floppy in the case of data failures on the boot disk. Adaptec 1540/1542 B and WD7000 are examples of such controllers.

B.4.2.2 Configurable Controllers

Configurable controllers give the administrator the ability to remap the `C:` and `D:` drives. Hence, a suggested configuration could be to use `vxrootmir` to mirror the boot disk onto another available disk and in case of a failure on the disk mapped to `C:`, remap the mirrored boot disk to `C:` and reboot the system. Such remapping typically requires a system floppy provided by the manufacturer of the controller. Examples are Adaptec 1740 and 1542 C.

B.4.2.3 Auto-failover Controllers

For controllers with auto-failover capabilities, at system boot, the controller searches a series of disk devices until it finds one that is available. The search pattern is SCSI ID 0, SCSI ID 1, and if these fail and a second controller is attached, SCSI ID 0 and SCSI ID 1 on the second controller. Note that it will never configure disks from separate controllers together as `C:` and `D:`; eg, if controller 1 disk 0 fails, it will map controller 1 disk 1 into `C:` and nothing from the second controller into `D:`.

The best choice for configuring a system in this case is to mirror the boot disk (SCSI ID 0 on the first controller) to SCSI ID 1 on the first controller. If multiple controllers are available, it is also possible to mirror the boot disk to SCSI ID 0 on the second controller. This gives the administrator the ability to have the system failover automatically even if all disks on the first controller become unavailable, such as due to cable/terminator failure or an electronic failure on the controller itself. The above applies only in the case where all the controllers attached to the system are auto-failover. Examples of auto-failover controllers are Adaptec 1742/1744 and DPT 2012B controllers.

B.4.2.4 Micro-channel (MCA) SCSI Adapter

The behavior of the MCA SCSI disk controller is quite different from the ones explained above. An MCA controller attempts to map disk with ID 0 to `C:`, if such a disk exists. Among other disks the disk with the lowest ID is mapped to `D:`. In the absence of a disk with ID 0, the controller maps the disk with the highest ID to `C:`, and the disk with the next highest ID to `D:`. It should also be noted that if a disk with ID 0 is present and fails due to an electronic or media failure error, the controller will not auto-failover to `D:`. The VxVM boot floppy should be used in order to boot from the desired disk.

B.4.3 Booting After Failures

While there are many types of failures that can prevent a system from booting (even with auto-failover controllers), the same basic procedure can be taken to bring the system up. When a system fails to boot, the administrator should first try to identify the failure by the evidence left behind on the screen, and repair it if possible (for example, if a drive was accidentally powered off). If the problem cannot be repaired (such as data errors on the boot disk), boot the system from a backup boot disk so that the damage can be repaired or the failing disk replaced.

If the controller fails to map any disks when the regular boot disk fails, rearrange the physical disk devices so that the backup boot disk is mapped into `C:`. If the controller has auto-failover capabilities, the system may manage to boot itself despite the errors. The administrator will usually find out about the failure because of mail received from the Volume Manager when it notices the failure.

If the controller does not have auto-failover capabilities, or if the failure was not detectable by the controller, the drive being mapped into `C:` by the controller is incapable of booting the system. The easiest way to boot the system in this situation is to boot using the VxVM boot floppy to specify a backup boot disk besides the one mapped into `C:`. To boot with the VxVM boot floppy, place the floppy in floppy drive `A:` and power up the machine. After system initialization, the administrator will see this prompt on the screen:

```
Enter kernel name [C:unix]:
```

The administrator should now enter the letter corresponding to the backup boot disk. This will depend on your configuration, as well as any auto-failover procedures taken by the controller. For example, with a simple controller the system has probably been configured so that the disk mapped into `D:` is the backup disk. In this case, the administrator would enter

```
D:
```

and press Return at the prompt to boot the system from the backup disk.

Note that auto-failover controllers can confuse the drive mappings. For example, take a three-disk system that has a simple auto-failover controller which maps the two disks with the lowest SCSI ID's into `C:` and `D:`. Normally,

a disk with SCSI ID of 0 will be mapped into `C:` and the disk with a SCSI ID of 1 will be mapped into `D:`. If the first has failed completely, the controller will map the disk with SCSI ID 1 into `C:` and the disk with SCSI ID 2 into `D:` If the system still failed to boot off `C:` (SCSI disk 1) and the boot disk is also mirrored to SCSI disk 2, the administrator would specify `D:` to boot off the third disk on the system.

If the disk specified to the VxVM boot floppy is not a valid boot disk, the boot program will print an error. For example, if the administrator specifies a disk that does not exist, the screen will show:In this case, the administrator should

```
hd:get_fs: Can't get hard disk drive parameters
Use Ctrl-Alt-Del to reboot
```

recheck his drive configuration and reboot the system in order to specify a different disk.

Most controllers that auto-failover will print diagnostics to the screen informing the kind of failure, and also the mapping being done. For example, the Adaptec 1740 family of controllers has output as shown below if the SCSI disk 0 fails to respond on the bus during the controller BIOS initialization:

```
Adaptec AHA-1740 BIOS vX.XX Copyright 1992, Adaptec Inc.
[ Standard Mode ] Target 0 - Device Not Found Target 1 - Drive C:
(80h)
```

The screen clears soon after this message appears.

B.4.4 Failures And Recovery Procedures

As mentioned earlier, there are several possible failures that can cause the system to fail to boot. This section outlines some of the possible failures and gives instructions on how to correct the problem.

B.4.4.1 Failures Finding the Boot Disk

Early in the boot process, immediately following system initialization, the screen will show something like this:

```
NO ROM BASIC
SYSTEM HALTED
```

This message screen varies widely between systems. This means that the system BIOS was unable to read the `fdisk boot` program from the boot drive. This can occur if no disk was mapped into `C:` by the controller, if the SCSI bus has locked up, or if the drive mapped into `C:` has no `fdisk boot` program on it.

Common causes for this problem are:

- The disk mapped into `C:` is not powered on.
- The SCSI bus is not terminated (in auto-failover controllers).
- All disks that are candidates for auto-failover are powered off.
- There is a controller failure of some sort.
- A disk is failing and locking the bus, preventing any disks from identifying themselves to the controller, making the controller assume that there are no disks attached.

The first step in diagnosing this problem is to check carefully that everything on the SCSI bus is in order. If disks are powered off or the bus is unterminated, correct the problem and reboot the system.

If one of the disks has failed, removed the disk from the bus and replace it.

If no hardware problems are found, the error is probably due to data errors on the disk mapped into `C:`. In order to repair this problem, attempt to boot the system from a backup boot disk. If your controller allows you to use the boot floppy and you are unable to boot from a backup boot disk, there is still some type of hardware problem. Similarly, if swapping the failed boot disk with a backup boot disk fails to allow the system to boot, this also indicates hardware problems.

B.4.4.2 Invalid fdisk Partition Data

The fdisk partition on a disk determines the disk partition from which the `boot` program should be read. (See the section entitled "Managing Storage Devices" in the *SVR4.2 Advanced System Administrator's Guide* and the `hd`(7) and `fdisk`(1m) man pages for more information on disk partitioning and `fdisk`.)

Normally, the boot disk will have one UNIX partition that is marked as active. If the `fdisk boot` program cannot find an active partition to boot from, it will display the following message:

```
Invalid Partition Table
```

The most likely reasons for this problem are: the `fdisk` program was used to mark the UNIX partition as no longer active, or the UNIX partition was deleted.

Boot the system from the alternate boot disk. Then use `fdisk` to look at the fdisk partitions. If the UNIX partition is not marked active, mark it active and save the changes. After you have marked the UNIX partition as active, try rebooting the system from the disk.

If there is no UNIX partition, you must re-add the disk. Refer to the section entitled "Re-adding a Failed Boot Disk."

B.4.4.3 Failure to Load the boot Program

If the `boot` program fails to load or start execution properly, the system will display:

```
Missing operating system
```

This can occur if data errors on the system have corrupted the `boot` program on disk, or perhaps the boot program was accidentally corrupted due to operator error. To see if the disk has failed generally, use the `vxdisk` command to list information about the disk. For example, if the failed disk is named `disk01`, execute the command

```
vxdisk list disk01
```

If the disk has failed, you will see the message:

```
vxdisk: Disk disk01: Not connected to a physical disk
```

This means that the underlying device for `disk01` has failed, so VxVM has stopped access to it through the volume manager disk devices. In this case, an attempt can be made to re-add the disk as described in the section “Re-adding A Failed Boot Disk”. If this fails, the disk must be replaced (see “Replacing A Failed Boot Disk”).

If the disk has not failed, the `vxdisk` command will produce a page of output. This means that the `boot` program on the disk was corrupted by a transient disk error, or was perhaps accidentally overwritten. If this is the case, an attempt can be made to rewrite the `boot` program to disk using the command:

```
/etc/vx/bin/vxbootsetup disk01
```

If this command fails, or if the console shows errors writing to the device, the disk should be replaced as described in the section entitled “Replacing A Failed Boot Disk”. If this command completes, but you continue to have problems with the drive, consider replacing it in any case.

B.4.4.4 Failures In Unix Partitioning

Once the `boot` program has loaded, it will attempt to access the boot disk through the normal UNIX partition information. If this information is damaged, the boot program will fail with the following error:

```
boot: No file system to boot from
```

If this message appears during the boot, the system should be booted from a backup boot disk. While booting, most disk drivers will display errors on the console about the invalid UNIX partition information on the failing disk. The messages will look similar to:

```
WARNING: Disk Driver: HA 0 TC 0 UNIX 0, Invalid disk VTOC
```

This indicates that the failure was due to an invalid disk partition. The administrator can attempt to re-add the disk as described in the section “Re-adding A Failed Boot Disk”; however, if the reattach fails, then the disk will need to be replaced as described in the section “Replacing A Failed Boot Disk”.

B.4.4.5 Failure To Find Files In /stand

Once the `boot` program has found a valid UNIX partition table, it will attempt to read and execute several files in the `/stand` file system. If it has any problem finding these files, it will output a message similar to:

```
boot: Cannot load <file>: file not found
```

where file can be one of `/etc/initprog/sip`, `/etc/initprog/mip`, or `unix`. The possible causes for these failures are that the slice containing the `/stand` file system does not exist, there is data corruption in the slice containing `/stand`, or that the files have actually been removed. If one of these files has been removed from the file system (and therefore would be removed in all copies of the `/stand` file system) the system is unbootable and irrecoverable; the system will need to be reinstalled. See the section “Recovering Data After Reinstallation”.

If the failure is due to data errors in the `/stand` file system, the system can be booted from a backup boot disk to investigate the problem. When the system boots, VxVM will notice errors on the stand volume and detach the mirror of the stand volume that resides on the failing boot disk. If the errors are correctable then VxVM will attempt to correct them and, if successful, the disk can continue to be used; otherwise the disk should be replaced as described in the section “Replacing A Failed Boot Disk”. To determine if the errors were corrected, print the information about the stand volume by issuing the command:

```
vxprint -tph -e 'assoc=="standvol"'
```

For example, if the failing boot disk was named `disk01` and the backup disk was named `disk02`, the output from `vxprint` will resemble the following:

```
PL   NAME         VOLUME      KSTATE     STATE     LENGTH   LAYOUT    NCOL/WIDTH MODE
SD   NAME         PLEX        DISK       DISKOFFS  LENGTH   [COL/]OFF FLAGS

pl   standvol-01 standvol    DETACHED   STALE     32768    CONCAT      -          RW
sd   disk01-01   standvol-01 0          0         32768    0           -
pl   standvol-02 standvol    ENABLED    ACTIVE    32768    CONCAT      -          RW
sd   disk02-03   standvol-02 0          0         32768    disk01    c0t1d0s0
```

Note the first mirror, standvol-01 has a kernel state (kstate) of DETACHED and a state of STALE. This is because the failures on the disk on which it resides, disk01, caused VxVM to remove the plex from active use in the volume. You can attempt to once again correct the errors by resynchronizing the failed mirror with the volume by issuing the command:

```
vxrecover standvol
```

If the Volume Manager fails to correct the error, the disk driver will print notices of the disk failure to the system console and the `vxrecover` utility will print an error message. If this occurs, the disk has persistent failures and should be replaced.

If the failure is due to the lack of a `/stand` slice, the system will boot normally and the `vxprint` command above will show all plexes with a KSTATE of ENABLED and a state of ACTIVE. The `vxbootsetup` utility can be used to attempt to fix this. For example, if the failed disk is `disk01`, then the command

```
/etc/vx/bin/vxbootsetup disk01
```

will attempt to correct the partitioning problems on the disk. If this command fails then the disk will need to be replaced as described in the section “Replacing A Failed Boot Disk”.

B.4.4.6 Missing root or swap Partitions

During the later stages of the boot process, the UNIX kernel will access the root and swap volumes. During this time, VxVM expects to be able to access the mirrors of these volumes on the boot disk as UNIX slices. If either of these slices are missing due to pilot error or corruption of the UNIX partition table, the VxVM module in the kernel will be unable to configure the root or swap volume and will print an error message. For example, if the root partition is missing from the boot disk, the following message will appear:

```
WARNING: vxvm: Can't open disk ROOTDISK in group ROOTDG.
```

If it is removable media (like a floppy), it may not be mounted or ready. Otherwise, there may be problems with the drive.

```
Kernel error code 19 WARNING: volroot.c: failed to open disk
ROOTDISK, error 19. WARNING: volroot.c: failed to set up the root
disk, error 19. PANIC: vfs_mountroot: cannot mount root
```

If this problem (or the corresponding problem involving the swap area) occurs, boot the system from a backup boot disk and use the `vxbootsetup` utility (as described above) to attempt to recreate the needed partitions. If this command fails, the failed disk will need to be replaced as described in "Replacing A Failed Boot Disk".

B.4.4.7 Stale or Unusable Plexes on Boot Disk

If a disk is unavailable when the system is running, any mirrors of volumes that reside on that disk will become stale, meaning the data on that disk is out of date relative to the other mirrors of the volume. During the boot process, the system accesses only one copy of the root and stand volumes (the copies on the boot disk) until a complete configuration for those volumes can be obtained. If it turns out that the mirror of one of these volumes that was used for booting is stale, the system must be rebooted from a backup boot disk that contains non-stale mirrors. This problem can occur, for example, if the system has an auto-failover controller and was booted with the original boot drive turned off. The system will boot normally, but the mirrors that reside on the unpowered disk will be stale. If the system reboots with the drive turned back on, the system will boot using those stale plexes.

Another possible problem can occur if errors in the VxVM headers on the boot disk prevents VxVM from properly identifying the disk. In this case, VxVM will be unable to know the name of that disk. This is a problem because mirrors are associated with disk names, and therefore any mirrors on that disk are unusable.

If either of these situations occurs, the VxVM utility `vxconfigd` will notice it when it is configuring system as part of the `init` processing of the boot sequence. It will output a message describing the error, describe what can be done about it, and halt the system. For example, if the mirror `rootvol-01` of the root volume `rootvol` on disk `disk01` of the system was stale, `vxconfigd` would print the following message:

```
vxvm:vxconfigd: Warning Plex rootvol-01 for root volume is stale
or unusable. vxvm:vxconfigd: Error: System boot disk does not
have a valid root plex Please boot from one of the following
disks: Disk: disk01 Device: c0t1d0s0 vxvm:vxconfigd: Error:
System startup failed The system is down.
```

This informs the administrator that the disk `disk01` contains usable copies of the root and stand mirrors and should be used for booting. This is the name of the system backup disk. When this message appears, the administrator should reboot the system from a backup boot disk.

Once the system has booted, the exact problem needs to be determined. If the mirrors on the boot disk were simply stale, they will be caught up automatically as the system comes up. If, on the other hand, there was a problem with the private area on the disk, the administrator will need to re-add or replace the disk.

If the mirrors on the boot disk were unavailable, the administrator should get mail from the VxVM utilities describing the problem. Another way to discover the problem is by listing the disks with the `vxdisk` utility. In the above example, if the problem is a failure in the private area of `disk01` (such as due to media failures or accidentally overwriting the VxVM private region on the disk), the command `vxdisk list` would show the following output:

```
DEVICE TYPE DISK GROUP STATUS - - disk01 rootdg failed was:
c0t0d0s0
```

B.4.5 Re-adding and Replacing Boot Disks

Normally, replacing a failed disk is as simple as putting a new disk somewhere on the controller and running the VxVM replace disk commands. It's even possible to move the data areas from that disk to available space on other disks, or to use a "hot spare" disk already on the controller to replace the failure. For data that is not critical for booting the system, it doesn't matter where the data is located. All data that is not boot critical is only accessed by the Volume Manager after the system is fully operational. The Volume Manager can find this data for you.

On the other hand, boot-critical data must be placed in specific areas on the bootable disks in order for the boot process to find it. The system PROM constrains the location of this data. Therefore, the process of replacing a boot disk is slightly more complex.

When a disk fails, there are two possible routes that can be taken to correct the action. If the error(s) are transient or correctable, then the same disk can be re-used. This is known as *re-adding* a disk. In some cases, actions such as reformatting a failed disk or simply doing a complete surface analysis to rebuild the alternate-sector mappings will be sufficient to make a disk re-usable and thus a candidate for re-addition. On the other hand, if the disk has truly failed, then it should be completely replaced.

B.4.5.1 Re-adding a Failed Boot Disk

Re-adding a disk is actually the same procedure as replacing the disk, except that the same physical disk is used. Normally, a disk that needs to be re-added has been *detached*, meaning that VxVM has noticed that the disk has failed and has ceased to access it. For example, take a system that has two disks, `disk01` and `disk02` which are normally mapped into the system configuration during boot as disks `c:` and `d:`, respectively. A failure has occurred to `disk01` that has caused the disk to become detached. This can be confirmed by listing the disks with the `vxdisk` utility, as in:

```
vxdisk list
```

This would result in the following output:

```
DEVICE      TYPE      DISK      GROUP      STATUS
c0t0d0s0    sliced    -         -          error
c0t1d0s0    sliced    disk02    rootdg     online
-           -         disk01    rootdg     failed was:c0t0d0s0
```

Notice that the disk `disk01` has no device associated with it, and has a status of `failed` with an indication of the device that it was detached from. It is also possible that the device `c0t0d0s0` would not be listed at all; this would occur if the disk failed totally and the disk controller did not notice it on the bus.

This is not necessarily always the case. For example, if the boot disk has uncorrectable failures associated with the UNIX partition table (such as a missing root partition that cannot be corrected), the output of the `vxdisk list` command resembles the following:

```
DEVICE      TYPE      DISK      GROUP      STATUS
c0t0d0s0    sliced    disk01    rootdg     online
c0t1d0s0    sliced    disk02    rootdg     online
```

However, because the error was not correctable by the described procedures, the disk is still deemed to have failed. In this case, it is necessary to detach the failing disk from it's device by hand. This is done using the "Remove Disk for Replacement" function of the `vxdiskadm` utility (See the `vxdiskadm`(1M) man page or the *VERITAS Volume Manager User's Guide* for more information about `vxdiskadm`).

Once the disk is detached from the device, any special procedures for correcting the problem can be taken, such as reformatting the device.

To re-add the disk, use the "Replace a failed or removed disk" function of the `vxdiskadm` utility to replace the disk, and select the same device as the replacement. In the above example, this would mean replacing `disk01` with the device `c0t0d0s0`.

If a re-add of the disk fails, the disk should be replaced.

B.4.5.2 Replacing a Failed Boot Disk

When a boot disk needs to be replaced, the system should first be booted off a backup boot disk and if the failing disk is not detached from it's device it should be manually detached using `vxdiskadm`. Once the disk is detached, the system should be shut down and the hardware replaced.

The replacement disk should have at least as much storage capacity as was in use on the disk being replaced. If the new disk has as much or more space than the old one, it can just be replaced. To determine the minimum size of a replacement disk you need to determine how much space was in use on the disk that failed.

The replacement disk should be large enough such that the region of the disk for storing subdisks will be large enough to hold all subdisks of the original disk at their current disk offsets.

To approximate the size of the replacement disk, use the command:

```
vxprint -st -e 'sd_disk="diskname"'
```

Add the values under the `DISKOFFS` and `LENGTH` columns for the last subdisk listed. The total is in 512-byte multiples. Divide the sum by 2 for the total in kilobytes.

Note – Disk sizes reported by manufacturers do not usually represent usable capacity. Also, some manufacturers report millions of bytes rather than megabytes, which are not equivalent.

Once a replacement disk has been found, shut down the machine cleanly and replace the necessary hardware. The replacement disk should be added at the same location on the bus (e.g., for SCSI controllers the disk should have the same SCSI ID) as the disk being removed.

When the hardware replacement is complete, boot the system and replace the failing disk with the new device that was just added using `vxdiskadm`.

B.5 Reinstallation Recovery

Occasionally, your system may need to be reinstalled after some types of failures. Reinstallation is necessary if all copies of your root (boot) disk are damaged, or if certain critical files are lost due to file system damage. When a

failure of either of these types occurs, you must reinstall the entire system, since there is currently no method of restoring the root file system from backup.

If these types of failures occur, you should attempt to preserve as much of the original Volume Manager configuration as possible. Any volumes not directly involved in the failure may be saved. You do not have to reconfigure any volumes that are preserved.

This section describes the procedures used to reinstall VxVM and preserve as much of the original configuration as possible after a failure.

B.5.1 General Recovery Information

System reinstallation completely destroys the contents of any disks that are reinstalled. Any VxVM-related information, such as data in the VxVM private areas on removed disks (containing the disk identifier and copies of the VxVM configuration), is removed during reinstallation. The removal of this information makes the disk unusable as a VxVM disk.

The system root disk is always involved in reinstallation. Other disks may also be involved. If the root disk was placed under Volume Manager control (either during Volume Manager installation or by later encapsulation), that disk and any volumes or volume mirrors on it are lost during reinstallation. In addition, any other disks that are involved in the reinstallation (or that are removed and replaced), may lose Volume Manager configuration data (including volumes and mirrors).

If a disk (including the root disk) is not under Volume Manager control prior to the failure, no Volume Manager configuration data is lost at reinstallation. Any other disks to be replaced can be replaced following the procedures in the *VERITAS Volume Manager User's Guide*. Although it simplifies the recovery process after reinstallation, not having the root disk under Volume Manager control increases the likelihood of a reinstallation being necessary. Having the root disk under VxVM control, and creating mirrors of the root disk contents, eliminates many of the problems that require system reinstallation.

When reinstallation is necessary the only volumes saved are those that reside on, or have copies on, disks that are not directly involved with the failure and reinstallation. Any volumes on the root disk and other disks involved with the failure and/or reinstallation are lost during reinstallation. If backup copies of

these volumes are available, the volumes can be restored after reinstallation. The exceptions are the `root`, stand, and `usr` file systems; these file systems cannot be restored from backup.

B.5.2 Reinstallation and Reconfiguration Procedures

To reinstall the system and recover the VxVM configuration, perform the following procedure:

1. Prepare the system for installation. This includes replacing any failed disks or other hardware, and detaching any disks not involved in the reinstallation.
2. Install the operating system. Do this by reinstalling the base system and any other non-VxVM packages.
3. Install VxVM. Add the VxVM package, but *do not* execute the `vxinstall` command.
4. Recover the VxVM configuration.
5. Cleanup the configuration. This includes restoring any information in volumes affected by the failure or reinstallation, and recreating system volumes (`root`, `swap`, etc.).

These steps are described in the following sections.

B.5.2.1 Preparing the System for Reinstallation

To prevent the loss of data on disks not involved in the reinstallation, you should only involve the root disk in the reinstallation procedure. It is recommended that any other disks (that contain volumes) be disconnected from the system before you start the reinstallation procedure. Disconnecting the other disks ensures that they are unaffected by the reinstallation. For example, if the operating system was originally installed with a home file system on the second drive, it may still be recoverable. Removing the second drive ensures that the home file system remains intact.

B.5.2.2 Reinstalling the Operating System

Once any failed or failing disks have been replaced and disks uninvolved with the reinstallation have been detached, reinstall the operating system as described in the manuals for your operating system. Install the basic operating system prior to installing VxVM.

While the operating system installation progresses, make sure no disks other than the root disk are accessed in any way. If anything is written on a disk other than the root disk, the Volume Manager configuration on that disk could be destroyed.

Note – Several of the *Automatic* options for installation access drives other than the root drive without requiring confirmation from the administrator. Therefore, it is advised that you disconnect all other disks from the system prior to installing the operating system.

B.5.2.3 Reinstalling the Volume Manager

The installation of the Volume Manager has two parts:

- Loading VxVM from floppy disks or cartridge tape.
- Initializing the Volume Manager.

If you wish to reconstruct the Volume manager configuration left on the non-root disks, *do not* initialize the Volume Manager after the reinstallation.

To reinstall the Volume Manager, follow the instructions for loading the Volume Manager (from CDROM, floppy disk or cartridge tape) in Chapter 1 of the *VERITAS Volume Manager User's Guide. Do not* initialize the Volume Manager with the `vxinstall` command. Instead, follow the instructions in the following section of this document.

B.5.2.4 Recovering the Volume Manager Configuration

Once the VxVM package has been loaded recover the Volume Manager configuration by doing the following:

1. Shut down the system.
2. Reattach the disks that were removed from the system.

3. Reboot the system.
4. When the system comes up, bring the system to single-user mode by entering the following command:

   ```
   shutdown -g0 -iS -y
   ```

   You will be asked for the System Administration password.
5. Enter the password and press Return to continue.
6. You need to remove some files involved with installation that were created when you loaded VxVM but are no longer needed. To do this, enter the command:

   ```
   rm -rf /etc/vx/bin/reconfig.d /etc/vx/bin/state.d
   rm -rf /etc/vx/bin/install-db
   ```
7. Once these files are removed, you must start some VxVM daemons. Start the daemons by entering the command:

   ```
   /sbin/vxiod set 2
   ```
8. Start the Volume Manager Configuration Daemon, `vxconfigd`, by entering the command:

   ```
   /sbin/vxconfigd -m disable
   ```
9. Initialize the `vxconfigd` daemon by entering:

   ```
   vxdctl init
   ```
10. Enable `vxconfigd` by entering:

    ```
    vxdctl enable
    ```

The configuration preserved on the disks not involved with the reinstallation has now been recovered. However, because the root disk has been reinstalled, it appears to the Volume Manager as a non-VxVM disk. Therefore, the configuration of the preserved disks does not include the root disk as part of the VxVM configuration. If the root disk of your system and any other disk involved in the reinstallation were not under Volume Manager control at the time of failure and reinstallation, then the reconfiguration is complete at this point. If any other disks containing volumes or volume mirrors are to be replaced, follow the replacement procedures in the *VERITAS Volume Manager User's Guide*. There are several methods available to replace a disk. Choose the method that you prefer.

If the root disk (or another disk) was involved with the reinstallation, any volume or volume mirrors on that disk (or other disks no longer attached to the system) are now inaccessible. If a volume had only one mirror, contained on a disk that was reinstalled, removed, or replaced, then the data in that volume is lost and must be restored from backup. In addition, the system's `root` file system, `swap` area, and stand area are *not* located on volumes any longer. To correct these problems, follow the instructions in the section, "Configuration Cleanup" on page B-31.

B.5.2.5 Configuration Cleanup

The following sections describe how to clean up the configuration of your system after reinstallation of the Volume Manager.

B.5.2.6 Rootability Cleanup

To begin the cleanup of the Volume Manager configuration, remove any volumes associated with rootability. This must be done if the root disk was under Volume Manager control. The volumes to remove are:

- `rootvol`, which contains the `root` file system
- `swapvol`, which contains the `swap` area
- standvol, which contains the stand file system

To remove the root volume, use the `vxedit` command, as follows:

```
vxedit -fr rm rootvol
```

Repeat the command, using `swapvol` and standvol in place of `rootvol`, to remove the `swap` and `stand` volumes.

B.5.2.7 Volume Cleanup

After completing the rootability cleanup, you must determine which volumes need to be restored from backup. The volumes to be restored include any of which all mirrors (all copies of the volume) reside on disks that have been reinstalled or removed. These volumes are invalid and must be removed, recreated, and restored from backup. If only some mirrors or a volume exist on reinitialized or removed disks, these mirrors must be removed. The mirrors can be readded later.

To restore the volumes, do the following:

1. Establish which VM disks have been removed or reinstalled, by entering the command:

   ```
   vxdisk list
   ```

The Volume Manager displays a list of system disk devices and the status of these devices. For example, for a reinstalled system with three disks and a reinstalled root disk, the output of the `vxdisk list` command is similar to this:

```
DEVICE     TYPE     DISK      GROUP    STATUS
c0t0d0s    slice    -         -        error
0s2        d        disk02    rootd    online
c0t2d0s    slice    disk03    g        online
0c0t2d0    d        disk01    rootd    failed was: c0t0d0s0
s0-        slice               g
           d                   rootd
           -                   g
```

The previous display shows that the reinstalled root device, `c0t0d0s0` is not recognized as a VM disk and is marked with a status of `error`. `disk02` and `disk03` were not involved in the reinstallation and are recognized by the Volume Manager and associated with their devices (`c0t1d0s0` and `c0t2d0s0`). The former `disk01`, the VM disk that had been associated with the replaced disk device, is no longer associated with the device (`c0t0d0s0`).

If there had been other disks (with volumes or volume mirrors on them) removed or replaced during reinstallation, these disks would also have a disk device in `error` state and a VM disk listed as not associated with a device.

2. Once you know which disks have been removed or replaced, all the mirrors on disks with a status of `failed` must be located. Enter the command:

   ```
   vxprint -sF "%name" -e'sd_disk = "<disk>"'
   ```

   where <*disk*> is the name of a disk with a `failed` status. Be sure to enclose the disk name in quotes in the command. Otherwise, the command will return an error message. The `vxprint` command returns a list of volumes that have mirrors on the failed disk. Repeat this command for every disk with a `failed` status.

3. Check the status of each volume. To print volume information, enter:

```
vxprint -th <volume_name>
```

where *volume_name* is the name of the volume to be examined.

The `vxprint` command displays the status of the volume, its plexes, and the portions of disks that make up those plexes. For example, a volume named `fnah` with only one plex resides on the reinstalled disk named `disk01`. The `vxprint -th` command, applied to the volume `fnah`, produces the following display

```
DG NAME         GROUP-ID
DM NAME         DEVICE    TYPE       PRIVLEN   PUBLEN   PUBPATH             FLAGS
V  NAME         USETYPE   KSTATE     STATE     LENGTH   READPOL   PREFPLEX
PL NAME         VOLUME    KSTATE     STATE     LENGTH   LAYOUT    NCOL/WIDTH MODE
SD NAME         PLEX      DISK       DISKOFFS  LENGTH   [COL/]OFF FLAGS

v  fnah         fsgen     DISABLED   ACTIVE    24000    SELECT     -
pl fnah-01      fnah      DISABLED   NODEVICE  24000    CONCAT     -          RW
sd disk01-04    fnah-01   disk01     245759    24000    0          -
```

4. The only mirror of the volume is shown in the line beginning with `pl`. The `STATE` field for the mirror named `fnah-01` is `NODEVICE`. The mirror has space on a disk that has been replaced, removed, or reinstalled. Therefore, the mirror is no longer valid and must be removed. Since `fnah-01` was the only mirror of the volume, the volume contents are irrecoverable except by restoring the volume from a backup. The volume must also be removed. If a backup copy of the volume exists, you can restore the volume later. Keep a record of the volume name and its length, you will need it for the backup procedure.

5. To remove the volume, use the `vxedit` command. To remove `fnah`, enter the command:

   ```
   vxedit -r rm fnah
   ```

It is possible that only part of a mirror is located on the failed disk. If the volume has a striped mirror associated with it, the volume is divided between several disk. For example, the volume named `foo` has one striped mirror, striped across three disks, one of which is the reinstalled disk `disk31`. The output of the `vxprint -th` command for `foo` returns:

```
DG NAME         GROUP-ID
DM NAME         DEVICE    TYPE      PRIVLEN  PUBLEN   PUBPATH             FLAGS
V  NAME         USETYPE   KSTATE    STATE    LENGTH   READPOL    PREFPLEX
PL NAME         VOLUME    KSTATE    STATE    LENGTH   LAYOUT     NCOL/WIDTH MODE
SD NAME         PLEX      DISK      DISKOFFS LENGTH   [COL/]OFF  FLAGS

v  foo          fsgen     DISABLED  ACTIVE   10240     SELECT    -
pl foo-01       foo       DISABLED  NODEVICE 10240     STRIPE    -         RW
sd pdisk33-07 foo-01      pdisk33   424144   10240     0         -
sd pdisk32-07 foo-01      pdisk32   620544   10240     0         -
pl foo-02       foo       DISABLED  NODEVICE 10240     CONCAT    -         RW
sd pdisk31-08 foo-01      pdisk31   262144   10240     0         -
```

The display shows three disks, across which the mirror `foo-01` is striped (the lines starting with `sd` represent the stripes). One of the stripe areas is located on a failed disk. This disk is no longer valid, so the mirror named `foo-01` has a state of `NODEVICE`. Since this is the only mirror of the volume, the volume is invalid and must be removed. If a copy of `foo` exists on the backup media, it can be restored later. Keep a record of the volume name and length of any volumes you intend to restore from backup.

6. Use the `vxedit` command to remove the volume, as described earlier.

   A volume that has one mirror on a failed disk may also have other mirrors on disks that are still valid. In this case, the volume does not need to be restored from backup, since the data is still valid on the valid disks. The

output of the `vxprint -th` command for a volume with one plex on a failed disk (`pdisk31`) and another plex on a valid disk (`pdisk32`) would look like this:

```
DG NAME         GROUP-ID
DM NAME         DEVICE    TYPE      PRIVLEN  PUBLEN   PUBPATH                   FLAGS
V  NAME         USETYPE   KSTATE    STATE    LENGTH   READPOL     PREFPLEX
PL NAME         VOLUME    KSTATE    STATE    LENGTH   LAYOUT      NCOL/WIDTH MODE
SD NAME         PLEX      DISK      DISKOFFS LENGTH   [COL/]OFF   FLAGS

v  foo          fsgen     DISABLED  ACTIVE   10240     SELECT        -
pl foo-01       foo       DISABLED  ACTIVE   10240     STRIPE        -            RW
sd pdisk32-07   foo-01    pdisk32   620544   10240     0             -
pl foo-02       foo       DISABLED  NODEVICE 10240     CONCAT        -            RW
sd pdisk31-08   foo-02    pdisk31   262144   10240     0             -
```

This volume has two plexes, `foo-01` and `foo-02`. The first plex, `foo-01`, does not use any space on the invalid disk, so it can still be used. The second plex, `foo-02`, uses space on the invalid disk, `disk01`, and has a state of `NODEVICE`. Plex `foo-02` must be removed. However, the volume still has one valid plex containing valid data. If the volume needs to be mirrored, another plex can be added later. Note the name of the volume if you wish to create another plex later.

7. To remove an invalid plex, the plex must be dissociated from the volume and then removed. This is done with the `vxplex` command. To remove the plex `foo-02`, enter the following command:

   ```
   vxplex -o rm dis foo-02
   ```

8. Once all the volumes have been cleaned up, you must clean up the disk configuration as described in the section “Disk Cleanup” on page B-36.

B.5.2.8 Disk Cleanup

Once all invalid volumes and volume plexes have been removed, the disk configuration can be cleaned up. Each disk that was removed, reinstalled, or replaced (as determined from the output of the `vxdisk list` command) must be removed from the configuration.

To remove the disk, use the `vxdg` command. To remove the failed `disk01`, enter:

```
vxdg rmdisk disk01
```

If the `vxdg` command returns an error message, some invalid volume mirrors exist. Repeat the processes described in "Volume Cleanup" on page B-31 until all invalid volumes and volume mirrors are removed.

B.5.2.9 Rootability Reconfiguration

Once all the invalid disks have been removed, the replacement or reinstalled disks can be added to Volume Manager control. If the root disk was originally under VxVM control (the `root` and `stand` file systems and the `swap` area were on volumes), or you now wish to put the root disk under VxVM control, add this disk first.

To add the root disk to Volume Manager control, use the Volume Manager Support Operations (`vxdiskadm`). Enter:

```
vxdiskadm
```

and select menu item `2`, `Encapsulate a disk`. Follow the instructions and encapsulate the root disk for the system. For more information see the *VERITAS Volume Manager User's Guide.*

When the encapsulation is complete, reboot the system to multi-user mode.

B.5.2.10 Final Reconfiguration

Once the root disk is encapsulated, any other disks that were replaced should be added using `vxdiskadm`. If the disks were reinstalled during the operating system reinstallation, they should be encapsulated; otherwise, simply add them.

Once all the disks have been added to the system, any volumes that were completely removed as part of the configuration cleanup can be recreated using their contents restored from backup. The volume recreation can be done using `vxassist` or the VERITAS Visual Administrator interface.

To recreate the volumes `fnah` and `foo` using the `vxassist` command, enter:

```
vxassist make fnah 24000 vxassist make foo 4224 layout=stripe
  nstripe=3
```

Once the volumes are created, they can be restored from backup using normal backup / restore procedures.

Any volumes that had plexes removed as part of the volume cleanup can have these mirrors recreated following the instructions for mirroring a volume for the interface (`vxassist` or VERITAS Visual Administrator) you choose in the *VERITAS Volume Manager User's Guide.*

To replace the plex removed from the volume `foo` using `vxassist`, enter:

```
vxassist mirror foo
```

Once you have restored the volumes and plexes lost during reinstallation, the recovery is complete and your system should be configured as it was prior to the failure.

Glossary

associate

The process of establishing a relationship between Volume Manager objects; for example, a subdisk that has been created and defined as having a starting point within a plex is referred to as being associated with that plex.

atomic operation

An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.

concatenation

A layout style characterized by subdisks that are arranged sequentially and contiguously.

configuration database

A set of records containing detailed information on existing Volume Manager objects (such as disk and volume attributes). A single copy of a configuration database is called a configuration copy.

device name

The name used to access a physical disk, such as c0t0d0s0. This is also referred to as the disk access name.

dirty region logging

The way the Volume Manager monitors and logs modifications to a mirror. A bit map of changed regions is kept in an associated subdisk called a log subdisk.

disk

A collection of read/write data blocks that are indexed and can be accessed fairly quickly. Each disk has a universally unique identifier.

disk access name

The name used to access a physical disk, such as c0t0d0s0. The term device name is also used to refer to the disk access name.

disk access records

Configuration records used to specify the access path to particular disks. Each disk access record contains a name, a type, and possibly some type-specific information, which is used by the Volume Manager in deciding how to access and manipulate the disk that is defined by the disk access record.

disk group

A collection of disks that share the same configuration database. The root disk group (rootdg) is a special private disk group that always exists.

disk ID

A universally unique identifier that is given to each disk and can be used to identify the disk, even if it is moved.

disk media name

A logical or administrative name chosen for the disk, such as disk03. The term disk name is also used to refer to the disk media name.

disk media record

A configuration record that identifies a particular disk, by disk ID, and gives that disk a logical (or administrative) name.

dissociate

Dissociating Volume Manager objects removes any link that exists between them; for example, dissociating a subdisk from a mirror removes the subdisk from the mirror and adds the subdisk to the free pool.

free space

An area of a disk under VxVM control that is not allocated to any subdisk or reserved for use by any Volume Manager object.

initial swap area

The first disk region used as a swap area for the kernel.

log subdisk

A subdisk that is used to store a dirty region log. See Dirty Region Logging.

mirror

A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms *mirror* and *plex* can be used synonymously.

mirroring

A technique involving the use of multiple mirrors to make duplicate (potentially backup) copies of the information contained in a volume.

object

An entity that is defined to and recognized internally by the Volume Manager. The VxVM objects are: volume, plex, subdisk, disk, and disk group. There are actually two types of disk objects—one for the physical aspect of the disk and the other for the logical aspect.

partition

The standard division of a physical disk device, as supported directly by the operating system and disk drives.

persistent state logging

A logging type that ensures that only active mirrors are used for recovery purposes and prevents failed mirrors from being selected for recovery.

plex

A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms *mirror* and *plex* can be used synonymously.

private region

A region of a physical disk used to store private, structured Volume Manager information. The *private region* contains a disk header, a table of contents, and a configuration database. The table of contents maps the contents of the disk. The disk header contains a disk ID. All data in the private region is duplicated for extra reliability.

public region

A region of a physical disk managed by the Volume Manager that contains available space and is used for allocating subdisks.

root configuration

The configuration database for the root disk group. This is special in that it always contains records for other disk groups, which are used for backup purposes only. It also contains disk records that define all disk devices on the system.

root disk group

A special private disk group that always exists on the system.

root file system

The initial file system mounted as part of the UNIX kernel startup sequence.

root partition

The disk region on which the root file system resides.

root volume

The VxVM volume that contains the root file system, if such a volume is designated by the system configuration.

spanning

A technique that permits a file or database too large to fit on a single disk to span across multiple physical disks.

sparse mirror

A mirror that is not as long as the volume or that has holes (regions of the mirror that don't have a backing subdisk).

stripe

Relatively small, equally-sized areas that are allocated alternately on the subdisks of each striped mirror.

striping

A technique that spreads data across several physical disks using stripes. The data is allocated alternately to the stripes within the subdisks of each mirror.

subdisk

The internal object representing a portion of a physical disk. Subdisks can be allocated to mirrors to form volumes.

swap area

A disk region used to hold copies of memory pages swapped out by the system pager process.

swap volume

A VxVM volume that is configured for use as a swap area.

transaction

A set of configuration changes that succeed or fail as a group, rather than individually. Transactions are used internally to maintain consistent configurations.

volboot file

A small file that is used to locate copies of the root configuration. The file may list disks that contain configuration copies in standard locations, and can also contain direct pointers to configuration copy locations. `volboot` is stored in a system-dependent location.

hostid

A string that identifies a host to the Volume Manager. The *hostid* for a host is stored in its `volboot` file, and is used in defining ownership of disks and disk groups.

volume

A virtual disk, representing an addressable range of disk blocks used by applications such as file systems or databases. A volume is a collection of from one to eight plexes.

volume configuration daemon (vxconfigd)

The Volume Manager utility that is the interface between the kernel `volconfig` device and the other Volume Manager utilities. `vxconfigd` is a necessary component which makes sure Volume Manager configuration requests do not interfere with each other.

volume configuration device

The volume configuration device (`/dev/vx/config`) is the interface through which all configuration changes to the volume device driver are performed.

volume event log

The volume event log device (`/dev/vx/event`) is the interface through which volume driver events are reported to the utilities.

volume device driver

The driver that forms the virtual disk drive between the application and the physical device driver level. The volume device driver is accessed through a virtual disk device node whose character device nodes appear in `/dev/vx/rdsk`, and whose block device nodes appear in `/dev/vx/dsk`.

Index

Spine for 2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up