Sun™ StorEdge™ Volume Manager 2.6 System Administrator's Guide

THE NETWORK IS THE COMPUTER™

A Sun Microsystems, Inc. Business
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
1 650 960-1300 fax 1 650 969-9131

Part No. 805-5706-10
Revision A, July 1998

Send comments about this document to: `smcc-docs@sun.com`

Please
Recycle

Contents

Preface ix

1. Introduction to the Volume Manager 1-1

1.1 Volume Manager Overview 1-1

1.1.1 How the Volume Manager Works 1-1

1.1.2 Physical Objects 1-2

1.1.3 Volume Manager Objects 1-3

1.1.4 Relationships Between VxVM Objects 1-9

1.1.5 Volume Manager RAID Implementations 1-10

1.1.6 Hot-Relocation 1-26

1.1.7 Volume Resynchronization 1-27

1.1.8 Dirty Region Logging 1-29

1.1.9 VxSmartSync Recovery Accelerator 1-30

1.1.10 Volume Manager Rootability 1-31

1.1.11 Volume Manager Daemons 1-33

1.1.12 Volume Manager Interfaces 1-35

1.2 Disk Array Overview 1-35

1.2.1 Redundant Arrays of Inexpensive Disks (RAID) 1-37

1.3 Dynamic Multipathing 1-41

1.3.1 Path Failover Mechanism 1-42

1.3.2 Load Balancing 1-42

1.3.3 Booting From DMP Devices 1-42

2. VxVM Performance Monitoring 2-1

2.1 Performance Guidelines 2-1

2.1.1 Data Assignment 2-1

2.1.2 Striping 2-2

2.1.3 Mirroring 2-3

2.1.4 Mirroring and Striping 2-6

2.1.5 Using RAID-5 2-7

2.1.6 Hot-Relocation 2-8

2.2 Performance Monitoring 2-9

2.2.1 Performance Priorities 2-9

2.2.2 Getting Performance Data 2-10

2.2.3 Using Performance Data 2-11

2.3 Tuning the Volume Manager 2-15

2.3.1 General Tuning Guidelines 2-15

2.3.2 Tunables 2-15

2.3.3 Tuning for Large Systems 2-22

3. Disk and Disk Group Administration 3-1

3.1 Standard Disk Devices 3-1

3.2 Disk Groups 3-2

3.3 Disk and Disk Group Utilities 3-3

3.4 Using Disks 3-4

3.4.1 Initializing and Adding Disks 3-4

3.4.2 Removing Disks 3-7

3.4.3 Moving Disks 3-9

3.5 Detecting and Replacing Failed Disks 3-9

3.5.1 Hot-Relocation 3-9

3.5.2 Detecting Failed Disks 3-12

3.5.3 Replacing Disks 3-15

3.5.4 Moving Relocated Subdisks 3-17
3.6 Working With Disk Groups 3-18
3.6.1 Creating a Disk Group 3-18
3.6.2 Using Disk Groups 3-18
3.6.3 Removing a Disk Group 3-19
3.6.4 Moving Disk Groups Between Systems 3-19
3.7 Using Special Devices 3-24
3.7.1 Using vxdisk for Special Encapsulations 3-25
3.7.2 Using vxdisk for RAM Disks 3-26
3.7.3 Using vxdisk For Displaying Multipathing Information 3-27

4. Volume Administration 4-1
4.1 VxVM Operations 4-1
4.2 VxVM Utility Descriptions 4-2
4.2.1 Using vxassist 4-2
4.2.2 Manipulating the Volume Configuration Daemon With vxdctl 4-6
4.2.3 Removing and Modifying Entities With vxedit 4-6
4.2.4 Creating VxVM Objects With vxmake 4-7
4.2.5 Correcting Plex Problems With vxmend 4-8
4.2.6 Performing Plex Operations With vxplex 4-8
4.2.7 Printing Configuration Information With vxprint 4-9
4.2.8 Performing Subdisk Operations With vxsd 4-9
4.2.9 Printing Volume Statistics With vxstat 4-9
4.2.10 Tracing Volume Operations With vxtrace 4-10
4.2.11 Performing Volume Operations With vxvol 4-10
4.3 Subdisk Operations 4-11
4.3.1 Creating Subdisks 4-11
4.3.2 Removing Subdisks 4-12
4.3.3 Displaying Subdisk Information 4-12
4.3.4 Associating Subdisks 4-13

4.3.5 Dissociating Subdisks 4-15
4.3.6 Changing Subdisk Information 4-16
4.3.7 Moving Subdisks 4-16
4.3.8 Splitting Subdisks 4-17
4.3.9 Joining Subdisks 4-17
4.4 Plex Operations 4-18
4.4.1 Creating Plexes 4-18
4.4.2 Performing Backups Using Mirroring 4-19
4.4.3 Associating Plexes 4-20
4.4.4 Dissociating and Removing Plexes 4-21
4.4.5 Displaying Plex Information 4-22
4.4.6 Changing Plex Attributes 4-22
4.4.7 Changing Plex Status: Detaching and Attaching Plexes 4-24
4.4.8 Moving Plexes 4-26
4.4.9 Copying Plexes 4-27
4.5 Volume Operations 4-27
4.5.1 Creating Volumes 4-28
4.5.2 Initializing Volumes 4-30
4.5.3 Estimating Maximum Volume Size 4-31
4.5.4 Removing Volumes 4-32
4.5.5 Displaying Volume Information 4-33
4.5.6 Changing Volume Attributes 4-34
4.5.7 Resizing Volumes 4-35
4.5.8 Starting and Stopping Volumes 4-38
4.5.9 Mirroring Existing Volumes 4-40
4.5.10 Volume Recovery 4-40
4.6 RAID-5 Volume Operations 4-41
4.6.1 RAID-5 Volume Layout 4-41
4.6.2 Creating RAID-5 Volumes 4-43
4.6.3 Initializing RAID-5 Volumes 4-45

4.6.4 Failures and RAID-5 Volumes 4-46
4.6.5 RAID-5 Recovery 4-48
4.6.6 Miscellaneous RAID-5 Operations 4-50
4.7 Performing Online Backup 4-57

A. Volume Manager Error Messages A-1
A.1 Logging Error Messages A-1
A.2 Volume Manager Configuration Daemon Error Messages A-3
A.2.1 vxconfigd Usage Messages A-3
A.2.2 vxconfigd Error Messages A-5
A.2.3 vxconfigd Fatal Error Messages A-30
A.2.4 vxconfigd Notice Messages A-33
A.2.5 vxconfigd Warning Messages A-36
A.3 Kernel Error Messages A-49
A.3.1 Kernel Notice Messages A-49
A.3.2 Kernel Warning Messages A-53
A.3.3 Kernel Panic Messages A-63

B. Recovery B-1
B.1 Protecting Your System B-1
B.2 The UNIX Boot Process B-2
B.2.1 Booting After Failures B-3
B.3 Possible Root (/), swap, and usr Configurations B-4
B.3.1 Repairing Root (/) or /usr File Systems on Volumes B-5
B.3.2 Backing Up and Restoring the Root File System B-8
B.4 Failures and Recovery Procedures B-9
B.4.1 Failures in UNIX Partitioning B-9
B.4.2 Failures Accessing the Boot Device B-10
B.4.3 Failures Due to Incorrect Entries in /etc/vfstab B-11
B.4.4 Failures Due to Missing or Damaged /etc/system B-12

B.4.5 Failures Due to Booting From Unusable or Stale Plexes B-14
B.5 Hot-Relocation and Boot Disk Failures B-16
B.6 Re-Adding and Replacing Boot Disks B-16
B.6.1 Re-Adding a Failed Boot Disk B-17
B.6.2 Replacing a Failed Boot Disk B-18
B.7 Reattaching Disks B-19
B.8 Reinstallation Recovery B-19
B.8.1 General Reinstallation Information B-20
B.8.2 Reinstallation and Reconfiguration Procedures B-20
B.9 Plex and Volume States B-29
B.9.1 Plex States B-30
B.9.2 The Plex State Cycle B-32
B.9.3 Plex Kernel State B-33
B.9.4 Volume States B-33
B.9.5 Volume Kernel State B-34

Preface

Sun™ StorEdge™ Volume Manager 2.6 System Administrator's Guide provides user information for the Sun StorEdge Volume Manager 2.6. These instructions are designed for an experienced system administrator.

Using UNIX Commands

This document does not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information:

- *Solaris 2.x Handbook for SMCC Peripherals*
- AnswerBook™ online documentation for the Solaris™ 2.x software environment
- Other software documentation that you received with your system

Typographic Conventions

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output.	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized. Command-line variable; replace with a real name or value.	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be `root` to do this. To delete a file, type `rm` *filename.*

Related Documentation

TABLE P-2 Related Documentation

Application	Title	Part Number
User information	*Sun StorEdge Volume Manager 2.6 User's Guide*	805-5705-xx
Installation	*Sun StorEdge Volume Manager 2.6 Installation Guide*	805-5707-xx
Release notes	*Volume Manager 2.6 Release Notes*	805-5708-xx
Storage Administrator GUI	*Sun StorEdge Volume Manager Storage Administrator 1.0 User's Guide*	805-5709-xx

Ordering Sun Documents

SunDocs[SM] is a distribution program for Sun Microsystems technical documentation. Contact SunExpress for easy ordering and quick delivery. You can find a listing of available Sun documentation on the World Wide Web.

TABLE P-3 SunExpress Contact Information

Country	Telephone	Fax
Belgium	02-720-09-09	02-725-88-50
Canada	1-800-873-7869	1-800-944-0661
France	0800-90-61-57	0800-90-61-58
Germany	01-30-81-61-91	01-30-81-61-92
Holland	06-022-34-45	06-022-34-46
Japan	0120-33-9096	0120-33-9097
Luxembourg	32-2-720-09-09	32-2-725-88-50
Sweden	020-79-57-26	020-79-57-27
Switzerland	0800-55-19-26	0800-55-19-27
United Kingdom	0800-89-88-88	0800-89-88-87
United States	1-800-873-7869	1-800-944-0661

World Wide Web: `http://www.sun.com/sunexpress/`

Sun Welcomes Your Comments

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If a card is not available, you can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback
  1-415-786-6443

CHAPTER 1

Introduction to the Volume Manager

This chapter provides detailed information about the Sun StorEdge Volume Manager. The first part of this chapter describes the Volume Manager and its features; the second part provides general information on disk arrays and Redundant Arrays of Inexpensive Disks (RAID).

1.1 Volume Manager Overview

The section describes how the Volume Manager works and the objects that VxVM manipulates. Various features of the Volume Manager are discussed later in this section.

1.1.1 How the Volume Manager Works

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 disk without interrupting applications. 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.

1.1.2 Physical Objects

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

- Physical disks
- Partitions

1.1.2.1 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.

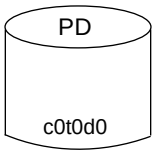


FIGURE 1-1 Example of a Physical Disk

1.1.2.2 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 can take up an entire physical disk, such as the partition shown in FIGURE 1-2.

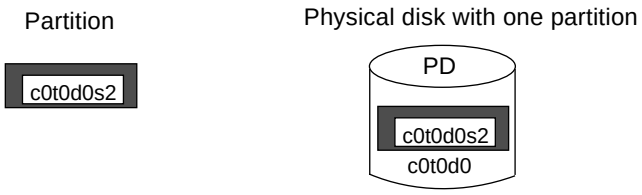


FIGURE 1-2 Example of a Partition

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

1.1.3 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

1.1.3.1 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. Each VM disk corresponds to at least one partition. A VM disk is typically composed of a *public region* (from which storage is allocated) and a *private region* (where configuration information is stored).

A VM disk is accessed using a unique *disk media name*, which you can supply (or else the Volume Manager assigns one that typically takes the form `disk##`). FIGURE 1-3 shows a VM disk with a disk media name of `disk01` that is assigned to the partition `c0t0d0s2`.

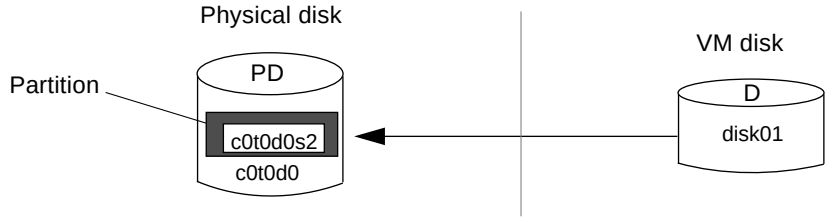


FIGURE 1-3 Example of a VM Disk

With the Volume Manager, applications access volumes (created on VM disks) rather than partitions.

1.1.3.2 Disk Groups

A *disk group* is a collection of VM disks that share a common configuration. A configuration consists of a set of records containing detailed information about existing Volume Manager objects, their attributes, and their relationships. The default disk group is `rootdg` (the root disk group). Additional disk groups can be created, as necessary. Volumes are created within a disk group; 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.

1.1.3.3 Subdisks

A *subdisk* is a set of contiguous disk blocks; subdisks are the basic units in which the Volume Manager allocates disk space. A VM disk can be divided into one or more subdisks. Each subdisk represents a specific portion of a VM disk, which is mapped to a specific region of a physical disk. 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`.

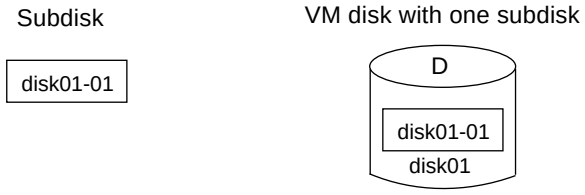


FIGURE 1-4 Example of a Subdisk

A VM disk may contain multiple subdisks, but subdisks cannot overlap or share the same portions of a VM disk. The example given in FIGURE 1-5 shows a VM disk, with three subdisks, that is assigned to one partition.

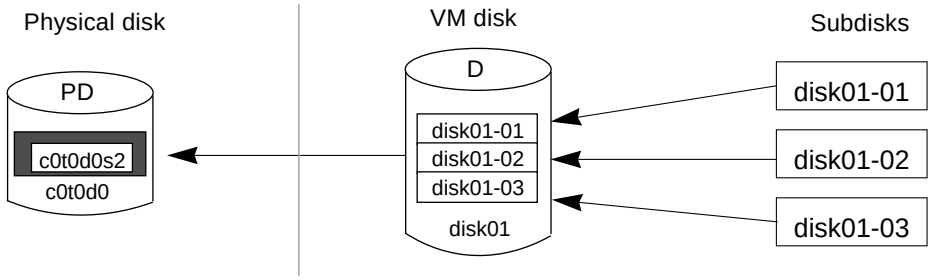


FIGURE 1-5 Example of Three Subdisks Assigned to One Partition

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

1.1.3.4 Plexes

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

- Concatenation
- Striping (RAID-0)
- RAID-5

Concatenation is discussed in this section. Details on striping (RAID-0) and RAID-5 are presented later in the chapter.

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.

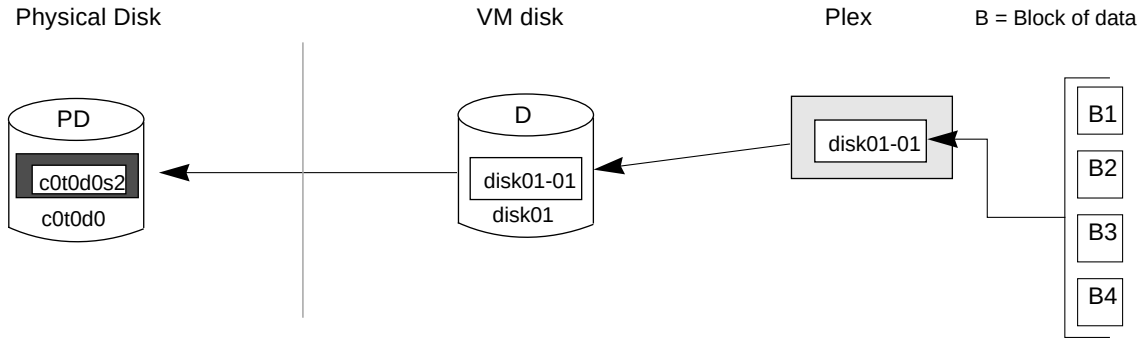


FIGURE 1-6 Example of Concatenation

Concatenation with multiple subdisks is useful when there is insufficient contiguous space for the plex on any one disk. Such concatenation can also be 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.

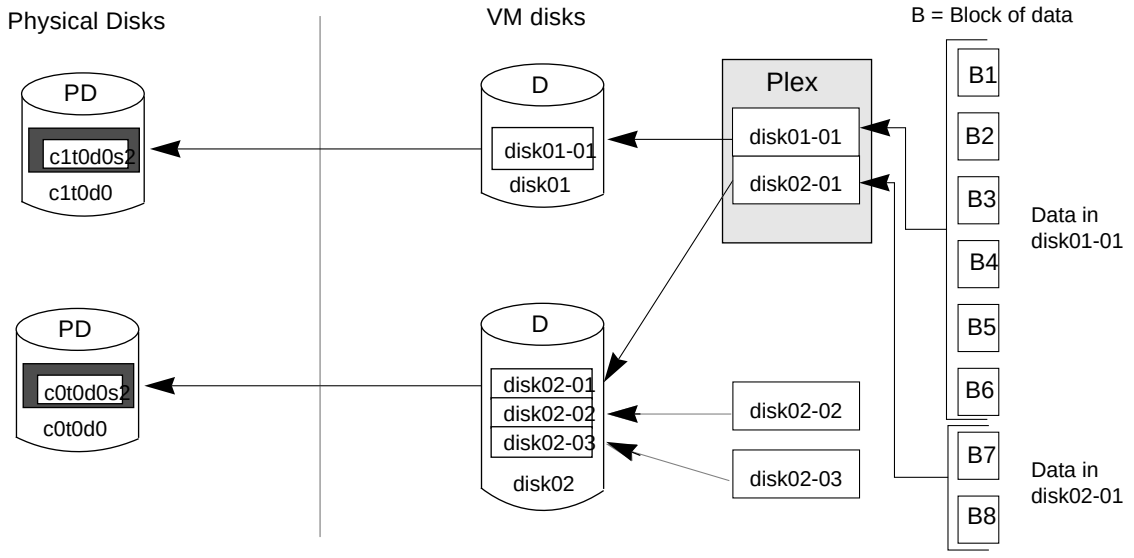


FIGURE 1-7 Example of Spanning

Since the first six blocks of data (`B1` through `B6`) consumed most or all of the room on the partition to which VM disk `disk01` is assigned, 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 to which VM disk `disk02` is assigned. That means that the remaining free space on VM disk `disk02` can be put to other uses. In this example, subdisks `disk02-02` and `disk02-03` are currently available for some other disk management tasks.

Caution – Spanning a plex across multiple disks increases the chance that a disk failure will result in failure of its volume. Use mirroring or RAID-5 (both described later) to substantially reduce the chance that a single disk failure will result in volume failure.

1.1.3.5 Volumes

A *volume* is a virtual disk device that appears to applications, databases, and file systems like a physical disk partition, but 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. For example, a volume can be mirrored on separate disks or moved to use different disk storage.

A volume can consist of up to 32 plexes, each of which contains one or more subdisks. In order for a volume to be usable, it must have at least one associated plex with at least one associated subdisk. Note that all subdisks within a volume must belong to the same disk group.

The Volume Manager uses the default naming conventions of `vol##` for volumes and `vol##-##` for plexes in a volume. Administrators are encouraged to select more meaningful names for their volumes.

A volume with one plex (FIGURE 1-8) contains a single copy of the data.

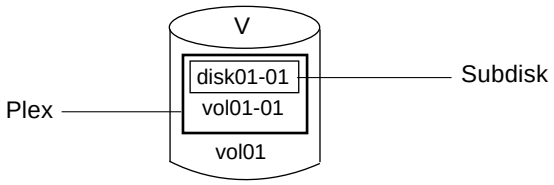


FIGURE 1-8 Example of a Volume with One Plex

Note that volume `vol01` in FIGURE 1-8 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 (see FIGURE 1-9) is considered “mirrored” and contains mirror images of the data. See Section 1.1.5.2, “Mirroring (RAID-1)” for more information on mirrored volumes.

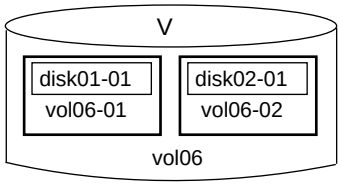


FIGURE 1-9 Example of a Volume with Two Plexes

Note that volume `vol06` in FIGURE 1-9 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 1-10 shows how a volume would look if it were set up with a simple, concatenated configuration.

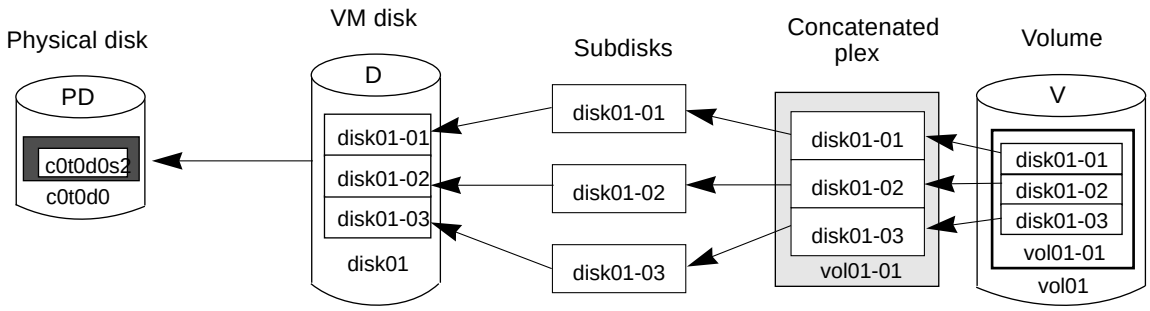


FIGURE 1-10 Example of a Volume in a Concatenated Configuration

1.1.4 Relationships Between VxVM Objects

Volume Manager objects are of little use until they are combined to build volumes. Volume Manager objects generally have the following relationship:

- VM disks are placed under VxVM control and grouped into disk groups
- One or more subdisks (each representing a specific portion of a disk) are combined to form plexes
- A volume is composed of one or more plexes

The example in FIGURE 1-11 illustrates the relationships between (virtual) Volume Manager objects, as well as how they relate to physical disks. This illustration shows a disk group containing two VM disks (`disk01` and `disk02`). `disk01` has a volume with one plex and two subdisks and `disk02` has a volume with one plex and a single subdisk.

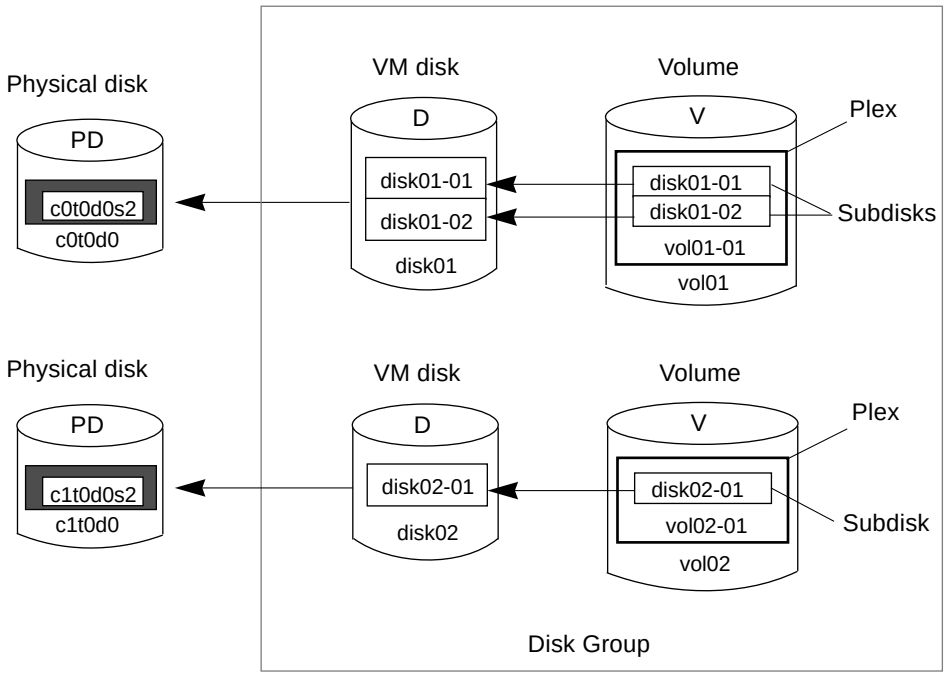


FIGURE 1-11 Relationships Between VxVM Objects

1.1.5 Volume Manager RAID Implementations

A *Redundant Array of Inexpensive Disks* (RAID) is a disk array (a group of disks that appear to the system as virtual disks, or volumes) that uses part of its combined storage capacity to store duplicate information about the data stored in the array. This duplicate information makes it possible to regenerate the data in the event of a disk failure.

This section focuses on the Volume Manager's implementations of RAID. For a general description of disk arrays and the various levels of RAID, refer to Section 1.2, "Disk Array Overview."

The Volume Manager supports the following levels of RAID:

- RAID-0 (striping)
- RAID-1 (mirroring)
- RAID-0 plus RAID-1 (striping and mirroring)
- RAID-5

The sections that follow describe how the Volume Manager implements each of these RAID levels.

1.1.5.1 Striping (RAID-0)

Striping is a technique of mapping data so that the data is interleaved among two or more physical disks. More specifically, a striped plex contains two or more subdisks, spread out over two or more physical disks. Data is allocated alternately and evenly to the subdisks of a striped plex.

The subdisks are grouped into “columns,” with each physical disk limited to one column. Each column contains one or more subdisks and can be derived from one or more physical disks. The number and sizes of subdisks per column can vary. Additional subdisks can be added to columns, as necessary.

Data is allocated in equal-sized units (called *stripe units*) that are interleaved between the columns. Each stripe unit is a set of contiguous blocks on a disk. The default stripe unit size is 64 kilobytes.

For example, if there are three columns in a striped plex and six stripe units, data is striped over three physical disks, as illustrated in FIGURE 1-12. The first and fourth stripe units are allocated in column 1; the second and fifth stripe units are allocated in column 2; and the third and sixth stripe units are allocated in column 3. Viewed in sequence, the first stripe begins with stripe unit 1 in column 1, stripe unit 2 in column 2, and stripe unit 3 in column 3. The second stripe begins with stripe unit 4 in column 1, stripe unit 5 in column 2, and stripe unit 6 in column 3. Striping continues for the length of the columns (if all columns are the same length) or until the end of the shortest column is reached. Any space remaining at the end of subdisks in longer columns becomes unused space.

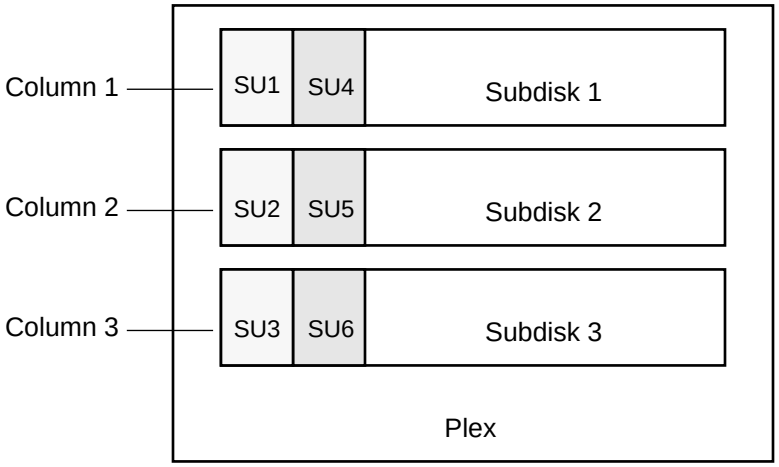


FIGURE 1-12 Striping Across Three Disks (Columns)

A *stripe* consists of the set of stripe units at the same positions across all columns. In FIGURE 1-12, stripe units 1, 2, and 3 constitute a single stripe.

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 helpful in balancing the I/O load from multi-user applications across multiple 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 or RAID-5 (both described later) to substantially reduce the chance that a single disk failure will result in failure of a large number of volumes.

FIGURE 1-13 shows a striped plex with three equal sized, single-subdisk columns. There is one column per physical disk.

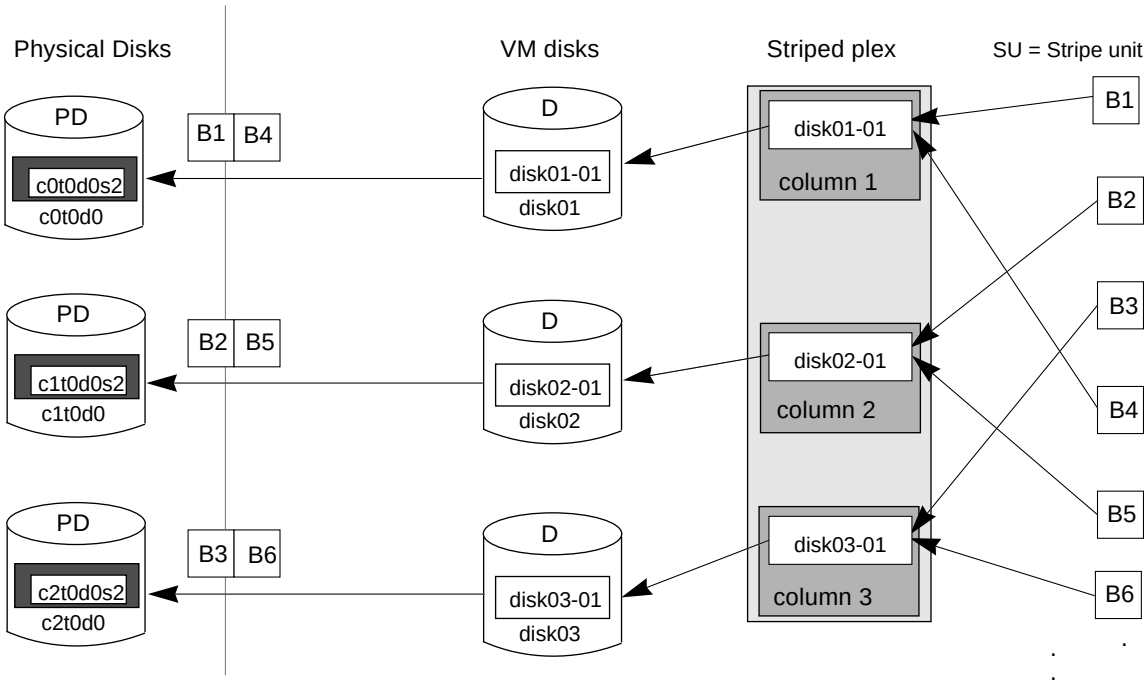


FIGURE 1-13 Example of a Striped Plex With One Subdisk per Column

Although the example in FIGURE 1-13 shows three subdisks that consume 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 1-14 shows a striped plex with 3 columns containing subdisks of different sizes. Each column 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).

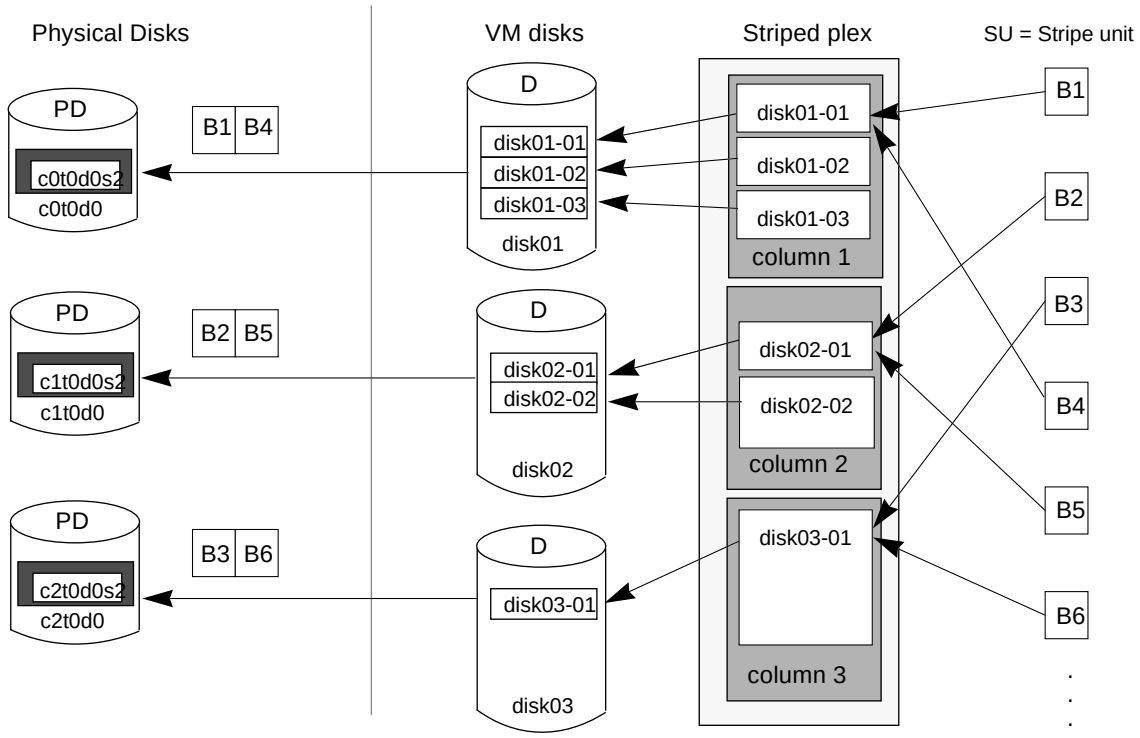


FIGURE 1-14 Example of a Striped Plex With Multiple Subdisks per Column

FIGURE 1-15 shows how a volume would look if it were set up for the simple striped configuration given in FIGURE 1-13.

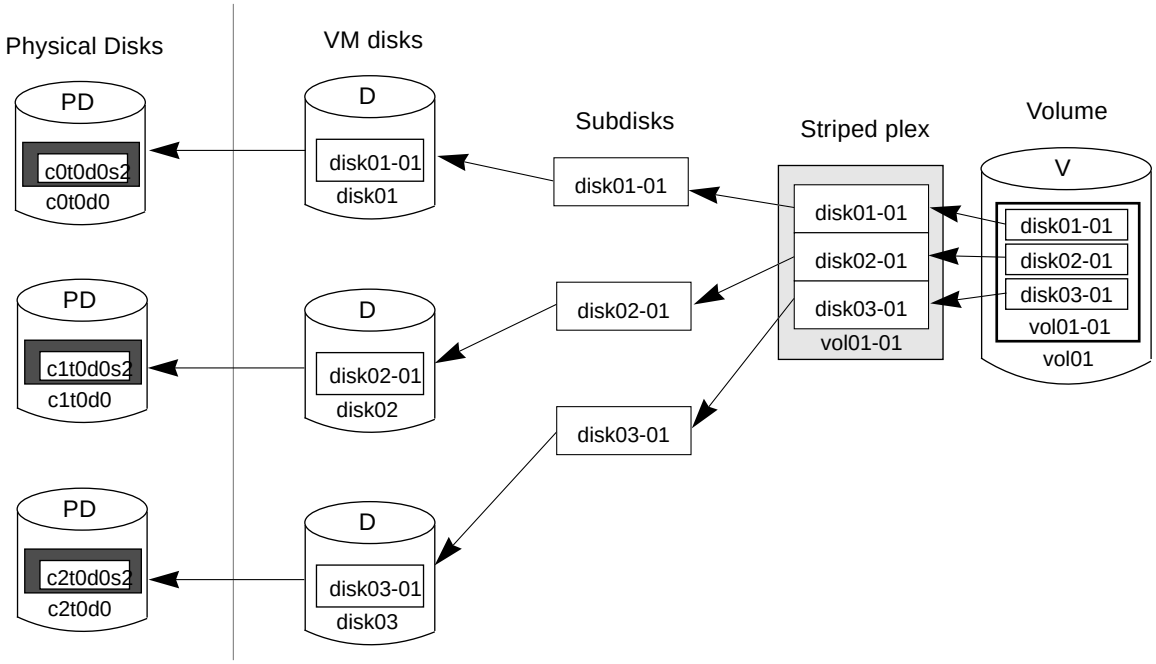


FIGURE 1-15 Example of a Volume in a Striped Configuration

1.1.5.2 Mirroring (RAID-1)

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 (and availability) of a striped or spanned volume.

1.1.5.3 Striping Plus Mirroring (RAID-0 + RAID-1)

The Volume Manager supports the combination of striping with mirroring. When used together on the same volume, striping plus mirroring offers the benefits of spreading data across multiple disks while providing redundancy of data.

For striping and mirroring to be effective together, the striped plex and its mirror must be allocated from separate disks. The layout type of the mirror can be concatenated or striped.

1.1.5.4 Volume Manager and RAID-5

This section describes the Volume Manager's implementation of RAID-5. For general information on RAID-5, refer to Section 1.2.1.6, "RAID-5."

Although both mirroring (RAID-1) and RAID-5 provide redundancy of data, their approaches differ. Mirroring provides data redundancy by maintaining multiple complete copies of a volume's data. Data being written to a mirrored volume is reflected in all copies. If a portion of a mirrored volume fails, the system will continue to utilize the other copies of the data.

RAID-5 provides data redundancy through the use of *parity* (a calculated value that can be used to reconstruct data after a failure). While data is being written to a RAID-5 volume, parity is also calculated by performing an *exclusive OR* (XOR) procedure on data. The resulting parity is then written to the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be re-created from the remaining data and the parity.

Traditional RAID-5 Arrays

A *traditional* RAID-5 array is made up of several disks organized in rows and columns, where a *column* is a number of disks located in the same ordinal position in the array and a *row* is the minimal number of disks necessary to support the full width of a parity stripe. FIGURE 1-16 shows the row and column arrangement of a traditional RAID-5 array.

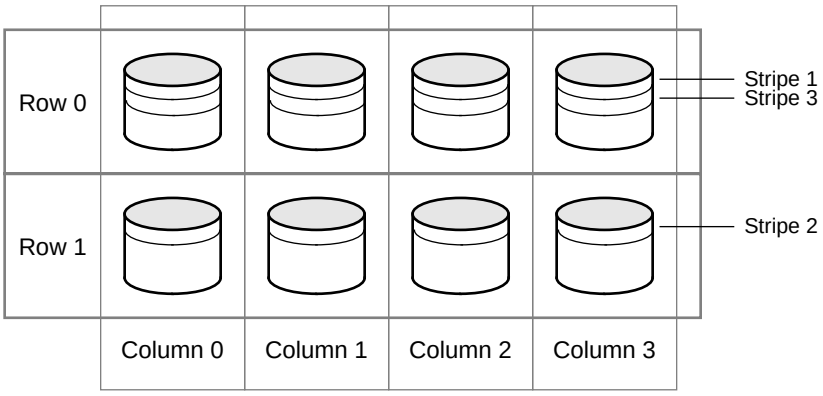


FIGURE 1-16 Traditional RAID-5 Array

This traditional array structure was developed to support growth by adding more rows per column. Striping is accomplished by applying the first stripe across the disks in Row 0, then the second stripe across the disks in Row 1, then the third stripe across Row 0's disks, and so on. This type of array requires all disks (partitions), columns, and rows to be of equal size.

VxVM RAID-5 Arrays

The Volume Manager's RAID-5 array structure differs from the traditional structure. Due to the virtual nature of its disks and other objects, the Volume Manager does not need to use rows. Instead, the Volume Manager uses columns consisting of variable length subdisks (as illustrated in FIGURE 1-17). Each subdisk represents a specific area of a disk.

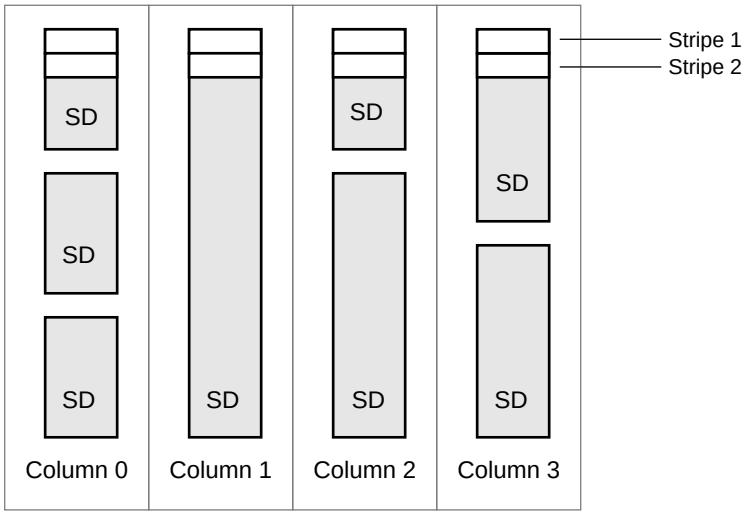


FIGURE 1-17 VxVM RAID-5 Array

With the Volume Manager RAID-5 array structure, each column can consist of a different number of subdisks and the subdisks in a given column can be derived from different physical disks. Additional subdisks can be added to the columns, as necessary. Striping (described in Section 1.1.5.1, “Striping (RAID-0)”) is accomplished by applying the first stripe across each subdisk at the top of each column, then another stripe below that, and so on for the entire length of the columns. For each stripe, an equal-sized stripe unit is placed in each column. With RAID-5, the default stripe unit size is 16 kilobytes.

Note – Mirroring of RAID-5 volumes is not currently supported.

Left-Symmetric Layout

There are several layouts for data and parity that can be used in the setup of a RAID-5 array. The layout selected for the Volume Manager’s implementation of RAID-5 is the left-symmetric layout. The left-symmetric parity layout provides optimal performance for both random I/Os and large sequential I/Os. In terms of performance, the layout selection is not as critical as the number of columns and the stripe unit size selection.

The left-symmetric layout stripes both data and parity across columns, placing the parity in a different column for every stripe of data. The first parity stripe unit is located in the rightmost column of the first stripe. Each successive parity stripe unit is located in the next stripe, left-shifted one column from the previous parity stripe unit location. If there are more stripes than columns, the parity stripe unit placement begins in the rightmost column again.

FIGURE 1-18 illustrates a left-symmetric parity layout consisting of five disks (one per column).

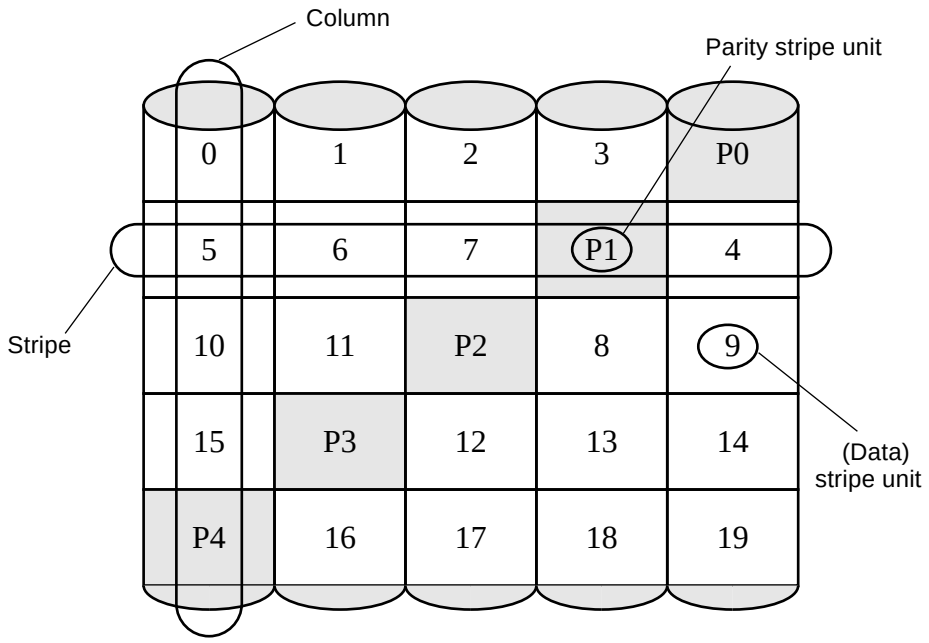


FIGURE 1-18 Left-Symmetric Layout

For each stripe, data is organized starting to the right of the parity stripe unit. In FIGURE 1-18, data organization for the first stripe begins at P0 and continues to stripe units 0-3. Data organization for the second stripe begins at P1, then continues to stripe unit 4, and on to stripe units 5-7. Data organization proceeds in this manner for the remaining stripes.

Each parity stripe unit contains the result of an exclusive OR (XOR) procedure performed on the data in the data stripe units within the same stripe. If data on a disk corresponding to one column is inaccessible due to hardware or software

failure, data can be restored by XORing the contents of the remaining columns' data stripe units against their respective parity stripe units (for each stripe). For example, if the disk corresponding to the leftmost column in FIGURE 1-18 were to fail, the volume would be placed in a degraded mode. While in degraded mode, the data from the failed column could be re-created by XORing stripe units 1-3 against parity stripe unit P0 to re-create stripe unit 0, then XORing stripe units 4, 6, and 7 against parity stripe unit P1 to recreate stripe unit 5, and so on.

Note – Failure of multiple columns in a plex with a RAID-5 layout will detach the volume. This means that the volume will no longer be allowed to satisfy read or write requests. Once the failed columns have been recovered, it might be necessary to recover the user data from backups.

Logging

Without logging, it is possible for data not involved in any active writes to be lost or silently corrupted if a disk fails and the system also fails. If this double-failure occurs, there is no way of knowing if the data being written to the data portions of the disks or the parity being written to the parity portions have actually been written. Therefore, the recovery of the corrupted disk may be corrupted itself.

Logging is used to prevent this corruption of recovery data. A log of the new data and parity is made on a persistent device (such as a disk-resident volume or non-volatile RAM). The new data and parity are then written to the disks.

In FIGURE 1-19, the recovery of Disk B is dependent on the data on Disk A and the parity on Disk C having both completed. The diagram shows a completed data write and an incomplete parity write causing an incorrect data reconstruction for the data on Disk B.

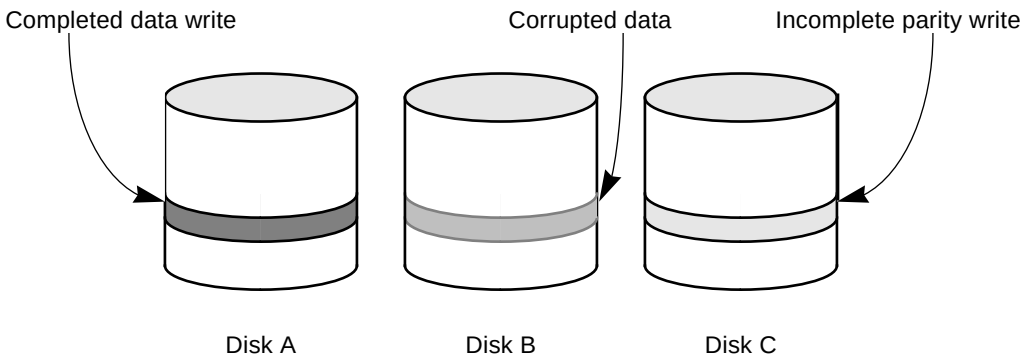


FIGURE 1-19 Incomplete Write

This failure case can be handled by logging all data writes before committing them to the array. In this way, the log can be replayed, causing the data and parity updates to be completed before the reconstruction of the failed drive takes place.

Logs are associated with a RAID-5 volume by being attached as additional, non-RAID-5 layout plexes. More than one log plex can exist per RAID-5 volume, in which case the log areas are mirrored.

Read-Modify-Write

When you write to a RAID-5 array, the following steps may be followed for each stripe involved in the I/O:

1. The data stripe units to be updated with new write data are accessed and read into internal buffers. The parity stripe unit is read into internal buffers.

2. The parity is updated to reflect the contents of the new data region. First, the contents of the old data undergo an exclusive OR (XOR) with the parity (logically removing the old data). The new data is then XORed into the parity (logically adding the new data). The new data and new parity are written to a log.

3. The new parity is written to the parity stripe unit. The new data is written to the data stripe units. All stripe units are written in a single write.

This process is known as a *read-modify-write* cycle, which is the default type of write for RAID-5. If a disk fails, both data and parity stripe units on that disk become unavailable. The disk array is then said to be operating in a *degraded* mode.

The read-modify-write sequence is illustrated in FIGURE 1-20.

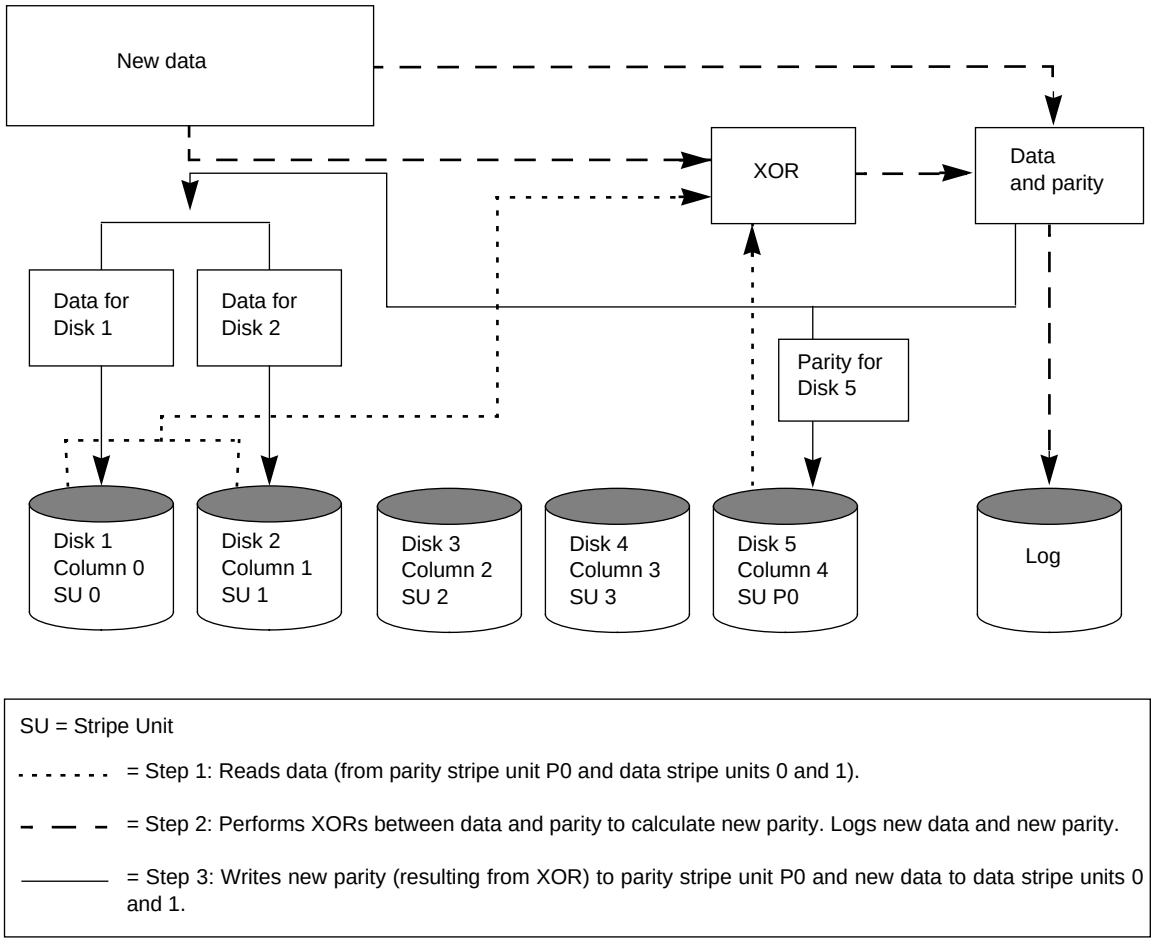


FIGURE 1-20 Read-Modify-Write

Full-Stripe Writes

When large writes (writes that cover an entire data stripe) are issued, the read-modify-write procedure can be bypassed in favor of a *full-stripe write.* A full-stripe write is faster than a read-modify-write because it does not require the read process to take place. Eliminating the read cycle reduces the I/O time necessary to write to the disk.

A full-stripe write procedure consists of the following steps:

1. All the new data stripe units are XORed together, generating a new parity value. The new data and new parity is written to a log.
2. The new parity is written to the parity stripe unit. The new data is written to the data stripe units. The entire stripe is written in a single write.

FIGURE 1-21 shows a full-stripe write.

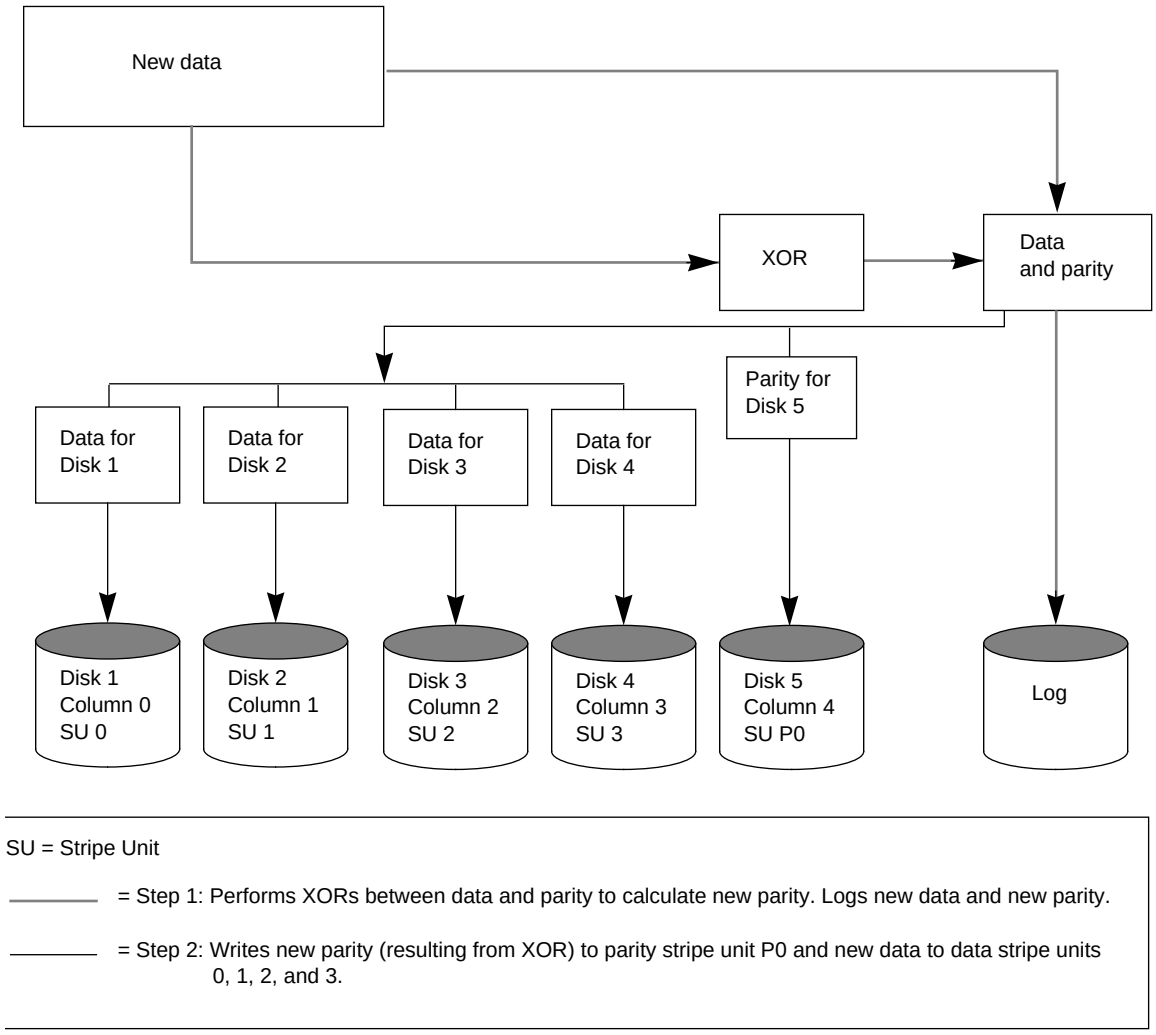


FIGURE 1-21 Full-Stripe Write

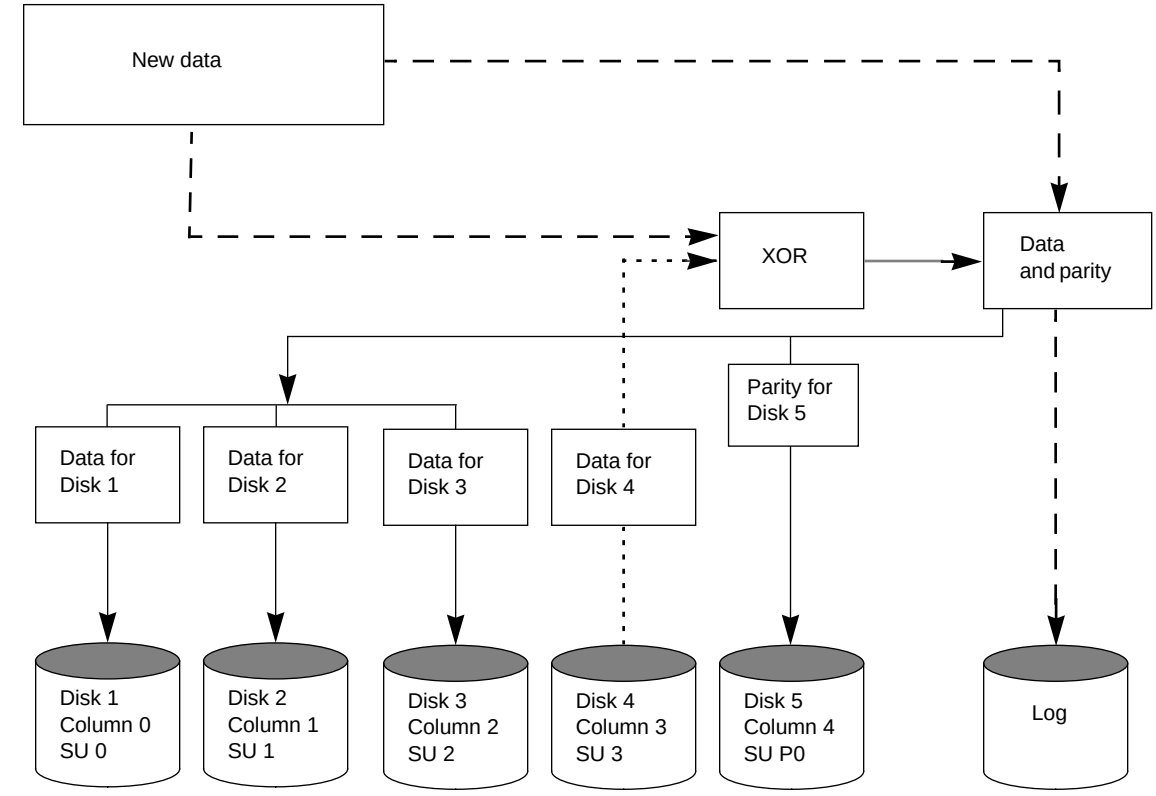
Reconstruct-Writes

When 50 percent or more of the data disks are undergoing writes in a single I/O, a *reconstruct-write* can be used. A reconstruct-write saves I/O time by XORing because it does not require a read of the parity region and only requires a read of the unaffected data (which amounts to less than 50 percent of the stripe units in the stripe).

A reconstruct-write procedure consists of the following steps:

1. Unaffected data is read from the unchanged data stripe unit(s).
2. The new data is XORed with the old, unaffected data to generate a new parity stripe unit. The new data and resulting parity are logged.
3. The new parity is written to the parity stripe unit. The new data is written to the data stripe units. All stripe units are written in a single write.

FIGURE 1-22 illustrates a reconstruct-write. A reconstruct-write is preferable to a read-modify-write in this situation because it reads only the necessary data disks, rather than reading the disks and the parity disk.



SU = Stripe Unit

- - - - - - - = Step 1: Reads data from unaffected data stripe unit 3.

– — – = Step 2: Performs XORs between old, unaffected data and new data. Logs new data and new parity.

———— = Step 3: Writes new parity (resulting from XOR) to parity stripe unit P0 and new data to data stripe units 0, 1, and 2.

FIGURE 1-22 Reconstruct-Write

1.1.6 Hot-Relocation

Hot-relocation is the ability of a system to automatically react to I/O failures on redundant (mirrored or RAID-5) VxVM objects and restore redundancy and access to those objects. The Volume Manager detects I/O failures on VxVM objects and relocates the affected subdisks to disks designated as *spare disks* and/or free space within the disk group. The Volume Manager then reconstructs the VxVM objects that existed before the failure and makes them redundant and accessible again.

When a partial disk failure occurs (that is, a failure affecting only some subdisks on a disk), redundant data on the failed portion of the disk is relocated and the existing volumes comprised of the unaffected portions of the disk remain accessible.

Note – Hot-relocation is only performed for redundant (mirrored or RAID-5) subdisks on a failed disk. Non-redundant subdisks on a failed disk are not relocated, but the system administrator is notified of their failure.

1.1.6.1 How Hot-Relocation Works

The hot-relocation feature is enabled by default. No system administrator intervention is needed to start hot-relocation when a failure occurs.

The hot-relocation daemon, `vxrelocd`, is responsible for monitoring VxVM for events that affect redundancy and performing hot-relocation to restore redundancy. `vxrelocd` also notifies the system administrator (via electronic mail) of failures and any relocation and recovery actions. See the `vxrelocd`(1M) manual page for more information on `vxrelocd`.

The `vxrelocd` daemon starts during system startup and monitors the Volume Manager for failures involving disks, plexes, or RAID-5 subdisks. When such a failure occurs, it triggers a hot-relocation attempt.

A successful hot-relocation process involves:

1. Detecting VxVM events resulting from the failure of a disk, plex, or RAID-5 subdisk.
2. Notifying the system administrator (and other users designated for notification) of the failure and identifying the affected VxVM objects. This is done through electronic mail.
3. Determining which subdisks can be relocated, finding space for those subdisks in the disk group, and relocating the subdisks. Notifying the system administrator of these actions and their success or failure.

4. Initiating any recovery procedures necessary to restore the volumes and data. Notifying the system administrator of the recovery's outcome.

Note – Hot-relocation does not guarantee the same layout of data or the same performance after relocation. The system administrator may therefore want to make some configuration changes after hot-relocation occurs.

1.1.6.2 How Space Is Chosen for Relocation

A spare disk must be initialized and placed in a disk group as a spare *before* it can be used for replacement purposes. If no disks have been designated as spares when a failure occurs, VxVM automatically uses any available free space in the disk group in which the failure occurs. If there is not enough spare disk space, a combination of spare space and free space is used. The system administrator can designate one or more disks as hot-relocation spares within each disk group. Disks can be designated as spares using the Visual Administrator, `vxdiskadm`, or `vxedit` (as described in later chapters). Disks designated as spares do not participate in the free space model and should not have storage space allocated on them.

When selecting space for relocation, hot-relocation preserves the redundancy characteristics of the VxVM object that the relocated subdisk belongs to. For example, hot-relocation ensures that subdisks from a failed plex are not relocated to a disk containing a mirror of the failed plex. If redundancy cannot be preserved using any available spare disks and / or free space, hot-relocation does not take place. If relocation is not possible, the system administrator is notified and no further action is taken.

When hot-relocation takes place, the failed subdisk is removed from the configuration database and VxVM takes precautions to ensure that the disk space used by the failed subdisk is not recycled as free space.

For information on how to take advantage of hot-relocation, refer to Chapter 2 and Chapter 3 of the *Sun StorEdge Volume Manager System Administrator's Guide*. Refer to the *Sun StorEdge Volume Manager Installation Guide* for information on how to disable hot-relocation.

1.1.7 Volume Resynchronization

When storing data redundantly using mirrored or RAID-5 volumes, the Volume Manager takes necessary measures to ensure that all copies of the data match exactly. However, under certain conditions (usually due to complete system failures), small amounts of the redundant data on a volume can become inconsistent or *unsynchronized*. Aside from normal configuration changes (such as detaching and

reattaching a plex), this can only occur when a system crashes while data is being written to a volume. Data is written to the mirrors of a volume in parallel, as is the data and parity in a RAID-5 volume. If a system crash occurs before all the individual writes complete, it is possible for some writes to complete while others do not, resulting in the data becoming unsynchronized. This is very undesirable. For mirrored volumes, it can cause two reads from the same region of the volume to return different results if different mirrors are used to satisfy the read request. In the case of RAID-5 volumes, it can lead to parity corruption and incorrect data reconstruction.

When the Volume Manager recognizes this situation, it needs to make sure that all mirrors contain exactly the same data and that the data and parity in RAID-5 volumes agree. This process is called *volume resynchronization*. For volumes that are part of disk groups that are automatically imported at boot time (such as `rootdg`), the resynchronization process takes place when the system reboots.

Not all volumes may require resynchronization after a system failure. Volumes that were never written or that were quiescent (that is., had no active I/O) when the system failure occurred could not have had any outstanding writes and thus do not require resynchronization. The Volume Manager notices when a volume is first written and marks it as *dirty*. When a volume is closed by all processes or stopped cleanly by the administrator, all writes will have completed and the Volume Manager removes the dirty flag for the volume. Only volumes that are marked dirty when the system reboots require resynchronization.

The exact process of resynchronization depends on the type of volume. RAID-5 volumes that contain RAID-5 logs can simply "replay" those logs. If no logs are available, the volume is placed in reconstruct-recovery mode and all parity is regenerated. For mirrored volumes, resynchronization is achieved by placing the volume in recovery mode (also called *read-writeback recovery mode*) and resynchronizing all data in the volume in the background. This allows the volume to be available for use while recovery is ongoing.

The process of resynchronization can be computationally expensive and can have a significant impact on system performance. The recovery process attempts to alleviate some of this impact by attempting to "spread out" recoveries to avoid stressing a specific disk or controller. Additionally, for very large volumes or for a very large number of volumes, the resynchronization process can take a long time. These effects can be addressed by using Dirty Region Logging for mirrored volumes, or by making sure that RAID-5 volumes have valid RAID-5 logs. For volumes used by database applications, the VxSmartSync™ Recovery Accelerator can be used (see Section 1.1.9, "VxSmartSync Recovery Accelerator").

1.1.8 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 and uses this information to recover only the portions of the volume that need to be recovered.

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 that are being written to. A dirty region log is maintained that contains a status bit representing each region of the volume. For any write operation to the volume, the regions being written are marked dirty in the log before the data is written. If a write causes a log region to become dirty when it was previously clean, the log is synchronously written to disk before the write operation can occur. On system restart, the Volume Manager will recover only those regions of the volume that are marked as dirty in the dirty region log.

Log subdisks are used to store the dirty region log of a volume that has DRL enabled. A volume with DRL has at least one log subdisk; multiple log subdisks can be used to mirror the dirty region log. Each log subdisk is associated with one of the volume's plexes. Only one log subdisk can exist per plex. If the plex contains only a log subdisk and no data subdisks, that plex can be referred to as a *log plex*. The log subdisk can also be associated with a regular plex containing data subdisks, in which case the log subdisk risks becoming unavailable in the event that the plex must be detached due to the failure of one of its data subdisks.

If the `vxassist` command is used to create a dirty region log, it creates a log plex containing a single log subdisk, by default. A dirty region log can also be created "manually" by creating a log subdisk and associating it with a plex. In the latter case, the plex may contain both a log subdisk and data subdisks.

Only a limited number of bits can be marked dirty in the log at any time. The dirty bit for a region is not cleared immediately after writing the data to the region. Instead, it remains marked as dirty until the corresponding volume region becomes the least recently used. If a bit for a given region is already marked dirty and another write to the same region occurs, it is not necessary to write the log to the disk before the write operation can occur.

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

1.1.9 VxSmartSync Recovery Accelerator

The VxSmartSync Recovery Accelerator for Mirrored Oracle® Databases is a collection of features designed to speed up the resynchronization process (also known as *resilvering*) for volumes used in conjunction with the Oracle Universal Database. These features employ an extended interface between VxVM volumes and the database software so that they can cooperate to avoid unnecessary work during mirror resynchronization. These extensions can result in an order of magnitude improvement in volume recovery times.

The only requirement to take advantage of VxSmartSync is to correctly configure the volumes. Once configured, no further user action is necessary. In the Volume Manager's view, there are two types of volumes used by the database:

- *redo log volumes* contain a database's redo logs
- *data volumes* are all other volumes used by the database (control files and tablespace files)

VxSmartSync handles these two types of volumes differently, and they must be configured correctly to take full advantage of the extended interfaces. The only difference between the two types of volumes is that redo log volumes should have dirty region logs, while data volumes should not.

1.1.9.1 Data Volume Configuration

The improvements in recovery time for data volumes are achieved by letting the database software decide exactly which portions of the volume require recovery. The database keeps its own logs of changes to the data in the database; therefore, it "knows" exactly which portions of the volume require recovery. The overall recovery time is significantly reduced by reducing the amount of space that requires recovery and enabling the database to control the recovery process. Additionally, the recovery takes place when the database software is started, not at system startup; this reduces the overall impact of recovery when the system reboots. Because the recovery is controlled by the database, the recovery time for the volume is exactly the resilvering time for the database (that is., the time required to replay the redo logs).

Because the database keeps its own logs, it is not necessary for VxVM to do any logging. Data volumes should therefore be configured as mirrored volumes *without* dirty region logs. In addition to improving recovery time, this avoids any run-time I/O overhead due to DRL, which improves normal database write access.

1.1.9.2 Redo Log Volume Configuration

A *redo log* is a log of changes to the database data. No logs of the changes to the redo logs are kept by the database, so the database itself cannot provide information about which sections require resilvering. Redo logs are also written sequentially, and since traditional dirty region logs are most useful with randomly-written data, they are of minimal use for reducing recovery time for redo logs. However, VxVM can significantly reduce the number of dirty regions by modifying the behavior of its Dirty Region Logging feature to take advantage of sequential access patterns. This decreases the amount of data needing recovery and significantly reduces recovery time impact on the system.

The enhanced interfaces for redo logs enable the database software to inform VxVM when a volume is to be used as a redo log. This enables VxVM to modify the volume's DRL behavior to take advantage of the access patterns. Since the improved recovery time depends on dirty region logs, redo log volumes should be configured as mirrored volumes *with* dirty region logs.

1.1.10 Volume Manager Rootability

The Volume Manager provides the capability of placing the root file system, `swap` device, and `usr` file system under Volume Manager control — this is called *rootability*. The *root disk* (that is, the disk containing the root file system) can be put under VxVM control through the process of *encapsulation*, which converts existing partitions on that disk to volumes. 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*; a volume that contains the root file system is referred to as a *root volume*.

It is possible to mirror the `rootvol` and `swapvol` volumes, as well as other parts of the root disk required for a successful boot of the system (such as `/usr`). This provides complete redundancy and recovery capability in the event of disk failure. Without Volume Manager rootability, the loss of the `root`, `swap`, or `usr` partition would prevent the system from being booted from surviving disks.

Mirroring disk drives critical to booting ensures that no single disk failure will leave the system unusable. Therefore, a suggested configuration would be to mirror the critical disk onto another available disk (using the `vxdiskadm` command). If the disk containing the `root` and `swap` partitions fails, the system can be rebooted from the disk containing the root mirror. For more information on mirroring the boot (root) disk and system recovery procedures, see the "Recovery" appendix in the *Sun StorEdge Volume Manager System Administrator's Guide*.

1.1.10.1 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 VxVM's `vxconfigd` daemon 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 `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 the primary plex (that is, the plex used for booting) of a `rootvol` or `swapvol` volume. Similarly, any mirrors of these volumes that might need to be used for booting cannot be striped or spanned.

1.1.10.2 Boot-Time Volume Restrictions

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

- The root volume (`rootvol`) must exist in the default disk group, `rootdg`. Although other volumes named `rootvol` may be created in disk groups other than `rootdg`, only the `rootvol` in `rootdg` can be used to boot the system.
- A `rootvol` volume has a specific minor device number: minor device 0. The `usr` volume does not have a specific minor device number.
- Restricted mirrors of `rootvol`, `var`, and `usr` 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`, `var`, and `usr` volumes are fully configured), the default volume configuration uses the overlay partition to access the data on the disk. (See Section 1.1.10.1, "Booting With Root Volumes.")
- Although it is possible to add a striped mirror to a `rootvol` device for performance reasons, you cannot stripe the primary plex or any mirrors of `rootvol` that may be needed for system recovery or booting purposes if the primary plex fails.
- `rootvol` and `swapvol` cannot be spanned or contain a primary plex with multiple non-contiguous subdisks.
- When mirroring parts of the boot disk, the disk being mirrored to must be large enough to hold the data on the original plex, or mirroring may not work.
- `rootvol` and `usr` cannot be DRL volumes.

In addition to these requirements, it is a good idea to have at least one contiguous mirror for each of the volumes for `root`, `usr`, `var`, `opt`, `varadm`, `usrkvm`, and `swap`. This makes it easier to convert these from volumes back to regular disk partitions (during an operating system upgrade, for example).

1.1.11 Volume Manager Daemons

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

- `vxconfigd`
- `vxiod`

1.1.11.1 The Volume Manager Configuration Daemon

The Volume Manager configuration daemon (`vxconfigd`) is responsible for maintaining Volume Manager disk and disk group configurations. `vxconfigd` communicates configuration changes to the kernel and modifies configuration information stored on disk. `vxconfigd` must be running before Volume Manager operations can be performed.

To Start the Volume Manager Configuration Daemon

`vxconfigd` is invoked by startup scripts during the boot procedure.

To determine whether the volume daemon is enabled, type 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, type:

```
vxdctl enable
```

The following message appears if `vxconfigd` is not running:

```
mode: not-running
```

If the latter message appears, start `vxconfigd` by typing:

```
vxconfigd
```

Once started, `vxconfigd` automatically becomes a background process.

By default, `vxconfigd` issues errors to the console. However, `vxconfigd` can be configured to log errors to a log file.

For more information on the `vxconfigd` daemon, refer to the `vxconfigd`(1M) and `vxdctl`(1M) manual pages.

1.1.11.2 The Volume I/O Daemon

The volume extended I/O daemon (`vxiod`) allows for some extended I/O operations without blocking calling processes.

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

To Start the Volume I/O Daemon

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

Verify that `vxiod` daemons are running by typing:

```
vxiod
```

Since `vxiod` is a kernel thread and is not visible to users via `ps`, this is the only way to see if any `vxiod`s are running.

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

```
10 volume I/O daemons running
```

where 10 is the number of `vxiod` daemons currently running.

If no `vxiod` daemons are currently running, start some by typing:

```
vxiod set 10
```

where 10 may be substituted by the desired number of `vxiod` daemons. It is generally recommended that at least one `vxiod` daemon exist per CPU.

1.1.12 Volume Manager 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 Manager 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 Volume Manager commands require a thorough understanding of Volume Manager concepts. Most Volume Manager commands require superuser privileges.
- Volume Manager Support Operations — The Volume Manager Support Operations interface (`vxdiskadm`) provides a menu-driven interface for performing disk and volume administration functions.

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

1.2 Disk Array Overview

This section provides an overview of traditional disk arrays.

Performing I/O to disks is a slow process because disks are physical devices that require time to move the heads to the correct position on the disk before reading or writing. If all of the read or write operations are done to individual disks, one at a time, the read-write time can become unmanageable. Performing these operations on multiple disks can help to reduce this problem.

A *disk array* is a collection of disks that appears to the system as one or more virtual disks (also referred to as *volumes*). The virtual disks created by the software controlling the disk array look and act (to the system) like physical disks. Applications that interact with physical disks should work exactly the same with the virtual disks created by the array.

Data is spread across several disks within an array, which allows the disks to share I/O operations. The use of multiple disks for I/O improves I/O performance by increasing the data transfer speed and the overall throughput for the array.

FIGURE 1-23 illustrates a standard disk array.

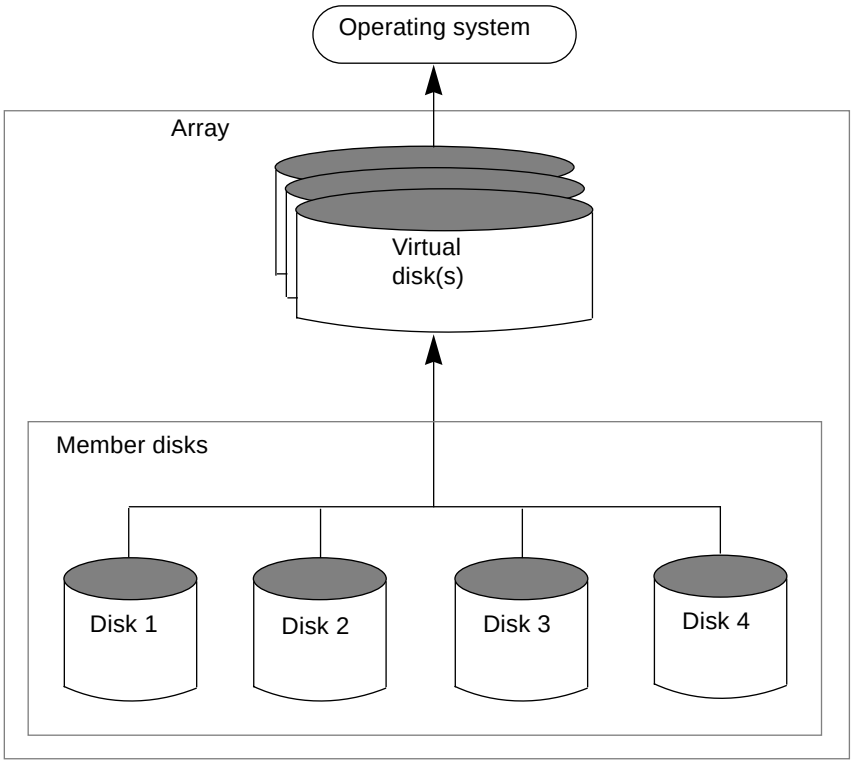


FIGURE 1-23 Standard Disk Array

1.2.1 Redundant Arrays of Inexpensive Disks (RAID)

A *Redundant Array of Inexpensive Disks* (RAID) is a disk array set up so that part of the combined storage capacity is used for storing duplicate information about the data stored in the array. The duplicate information enables you to regenerate the data in case of a disk failure.

Several levels of RAID exist. They are introduced in the following sections.

Note – The Volume Manager supports RAID levels 0, 1, and 5 only.

For information on the Volume Manager's implementations of RAID, refer to Section 1.1.5, "Volume Manager RAID Implementations."

1.2.1.1 RAID-0

Although it does not provide redundancy, striping is often referred to as a form of RAID, known as RAID-0. The Volume Manager's implementation of striping is described in Section 1.1.5.1, "Striping (RAID-0)." RAID-0 offers a high data transfer rate and high I/O throughput, but suffers lower reliability and availability than a single disk.

1.2.1.2 RAID-1

Mirroring is a form of RAID, known as RAID-1. The Volume Manager's implementation of mirroring is described in Section 1.1.5.2, "Mirroring (RAID-1)." Mirroring uses equal amounts of disk capacity to store the original plex and its mirror. Everything written to the original plex is also written to any mirrors. RAID-1 provides redundancy of data and offers protection against data loss in the event of physical disk failure.

1.2.1.3 RAID-2

RAID-2 uses bitwise striping across disks and uses additional disks to hold Hamming code check bits. RAID-2 is described in a University of California at Berkeley research paper entitled *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, by David A. Patterson, Garth Gibson, and Randy H. Katz (1987).

RAID-2 deals with error detection, but does not provide error correction. RAID-2 also requires large system block sizes, which limits its use.

1.2.1.4 RAID-3

RAID-3 uses a *parity disk* to provide redundancy. RAID-3 distributes the data in stripes across all but one of the disks in the array. It then writes the parity in the corresponding stripe on the remaining disk. This disk is the parity disk.

FIGURE 1-24 illustrates a RAID-3 disk array.

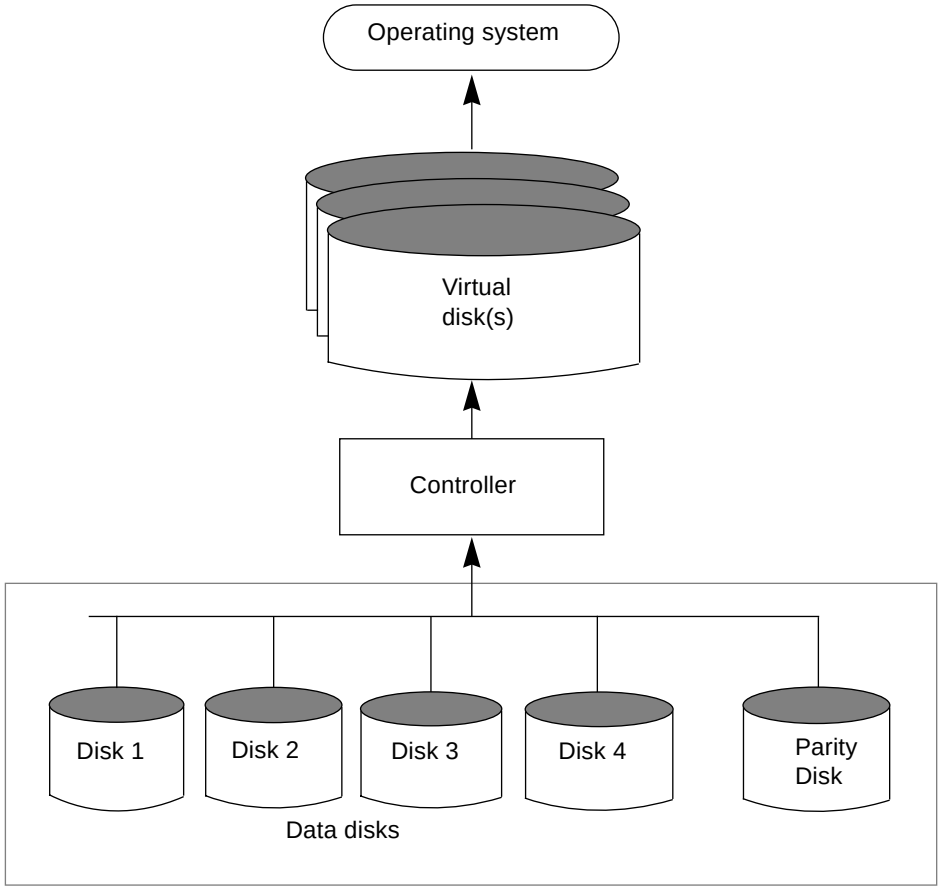


FIGURE 1-24 RAID-3 Disk Array

The user data is striped across the data disks. Each stripe on the parity disk contains the result of an exclusive OR (XOR) procedure done on the data in the data disks. If the data on one of the disks is inaccessible due to hardware or software failure, data can be restored by XORing the contents of the remaining data disks with the parity disk. The data on the failed disk can be rebuilt from the output of the XOR process.

RAID-3 typically uses a very small stripe unit size (also historically known as a *stripe width*), sometimes as small as one byte per disk (which requires special hardware) or one sector (block) per disk.

FIGURE 1-25 illustrates a data write to a RAID-3 array.

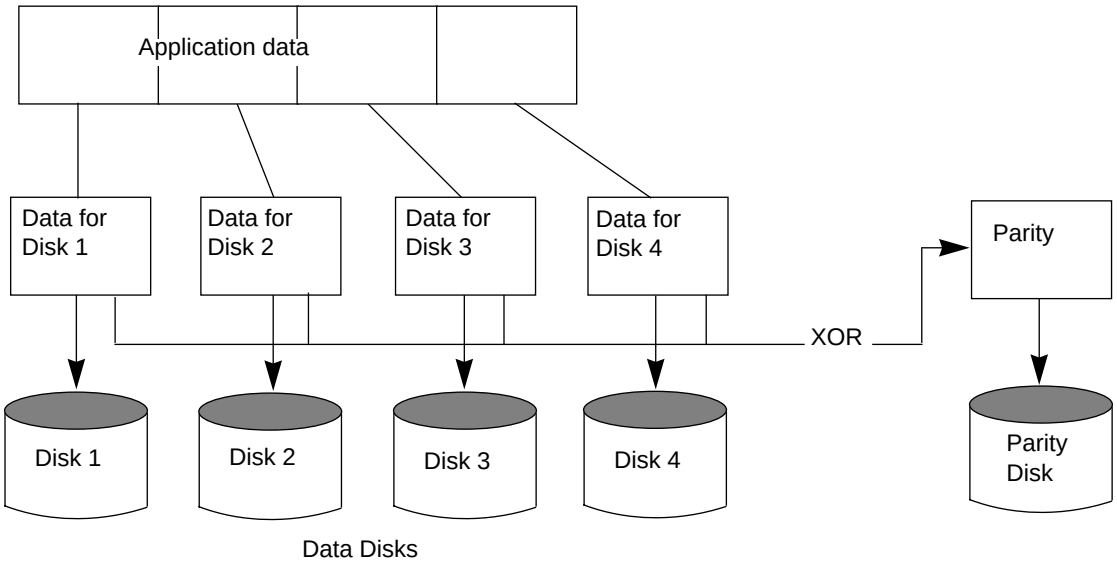


FIGURE 1-25 Data Writes to RAID-3

The parity disk model uses less disk space than mirroring, which uses equal amounts of storage capacity for the original data and the copy.

The RAID-3 model is often used with synchronized spindles in the disk devices. This synchronizes the disk rotation, providing constant rotational delay. This is useful in large parallel writes.

RAID-3 type performance can be emulated by configuring RAID-5 (described later) with very small stripe units.

1.2.1.5 RAID-4

RAID-4 introduces the use of independent-access arrays (also used by RAID-5). With this model, the system does not typically access all disks in the array when executing a single I/O procedure. This is achieved by ensuring that the stripe unit size is sufficiently large that the majority of I/Os to the array will only affect a single disk (for reads).

An array attempts to provide the highest rate of data transfer by spreading the I/O load as evenly as possible across all the disks in the array. In RAID-3, the I/O load is spread across the data disks, as shown in FIGURE 1-25, and each write is executed on all the disks in the array. The data in the data disk is XORed and the parity is written to the parity disk.

RAID-4 maps data and uses parity in the same manner as RAID-3, by striping the data across all the data disks and XORing the data for the information on the parity disk. The difference between RAID-3 and RAID-4 is that RAID-3 accesses all the disks at one time and RAID-4 accesses each disk independently. This enables the RAID-4 array to execute multiple I/O requests simultaneously (provided they map to different member disks), while RAID-3 can only execute one I/O request at a time.

RAID-4's read performance is much higher than its write performance. It performs well with applications requiring high read I/O rates. RAID-4 performance is not as high in small, write-intensive applications.

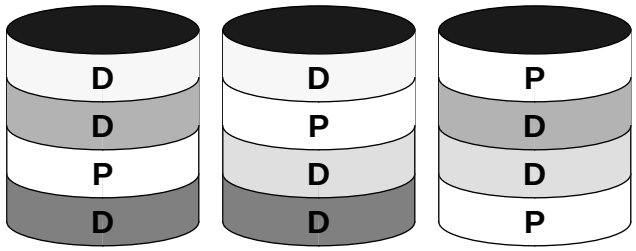
The parity disk can cause a bottleneck in the performance of RAID-4. This is because all the writes that are taking place simultaneously on the data disks must each wait its turn to write to the parity disk. The transfer rate of the entire RAID-4 array in a write-intensive application is limited to the transfer rate of the parity disk.

Since RAID-4 is limited to parity on one disk only, it is less useful than RAID-5.

1.2.1.6 RAID-5

RAID-5 is similar to RAID-4, using striping to spread the data over all the disks in the array and using independent access. However, RAID-5 differs from RAID-4 in that the parity is striped across all the disks in the array, rather than being concentrated on a single parity disk. This breaks the write bottleneck caused by the single parity disk write in the RAID-4 model.

FIGURE 1-26 illustrates parity locations in a RAID-5 array configuration. Every stripe has a column containing a parity stripe unit and columns containing data. The parity is spread over all of the disks in the array, reducing the write time for large independent writes because the writes do not have to wait until a single parity disk can accept the data.



D = Data stripe unit
P = Parity stripe unit

FIGURE 1-26 Parity Locations in a RAID-5 Model

For additional information on RAID-5 and how it is implemented by the Volume Manager, refer to Section 1.1.5.4, “Volume Manager and RAID-5.”

1.3 Dynamic Multipathing

The Volume Manager actively supports multiported disk arrays. It automatically recognizes multiple I/O paths to a particular disk device within the disk array. The Dynamic Multipathing feature of the Volume Manager provides greater reliability by providing a path failover mechanism. In the event of a loss of one connection to a disk, the system continues to access the critical data over the other healthy connections to the disk. The multipathing functionality also provides greater I/O throughput by balancing the I/O load uniformly across multiple I/O paths to the disk device.

In the Volume Manager all the physical disks connected to the system are represented as metadevices with one or more physical access paths. A single physical disk connected to the system is represented by a metadevice with one path, whereas a disk that is part of a disk array is represented by a metadevice that has two physical access paths. The user can use the Volume Manager administrative utilities such as `vmdisk` to display all the paths of a metadevice and the status information of the various paths.

1.3.1 Path Failover Mechanism

The Dynamic Multipathing feature in the Volume Manager enhances the system reliability when used with multiported disk arrays. In the event of a loss of one connection to the disk array, the multipathing module automatically selects the next active I/O paths for the I/O requests dynamically, without the administrator's intervention.

It also provides reconfiguration capability which allows the administrator to indicate to the DMP subsystem in the Volume Manager, if the connection is repaired or restored. The reconfiguration procedure also allows the detection of newly added devices, as well as devices that are removed after the system was fully booted.

1.3.2 Load Balancing

The Dynamic Multipathing feature also improves I/O throughput by balancing the load across multiple paths of a disk array.

1.3.3 Booting From DMP Devices

When the root disk is placed under the Volume Manager control, it is automatically accessed as a DMP device with one path if it is a single disk, or with more paths if the disk is part of a multiported disk array. By encapsulating the root disk, the system reliability is enhanced against loss of one or more of the existing physical paths to a disk.

CHAPTER 2

VxVM Performance Monitoring

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

2.1 Performance Guidelines

This section provides some guidelines on how to take advantage of various Volume Manager features. VxVM provides flexibility in configuring storage to improve system performance. Two basic strategies are available for optimizing performance:

- Assigning data to physical drives 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 striping and mirroring

VxVM also provides data redundancy (through mirroring and RAID-5), for continuous access to data in the event of some sort of disk failure.

2.1.1 Data Assignment

When deciding where to locate file systems, a system administrator typically attempts to balance I/O load among available disk 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.

2.1.2 Striping

Striping is a way of "slicing" data and storing it across multiple devices in order to improve access performance. Striping can provide increased access bandwidth for a plex. Striped plexes exhibit improved access performance for both read and write operations.

If the most heavily accessed volumes (containing file systems or databases) are identified, striping this "high traffic" data across portions of multiple disks, can increase access bandwidth to this data.

FIGURE 2-1 is an example of a single volume (`Hot Vol`) that has been identified as being a data access bottleneck. This volume is striped across four disks, leaving the remainder of those four disks free for use by less heavily used volumes.

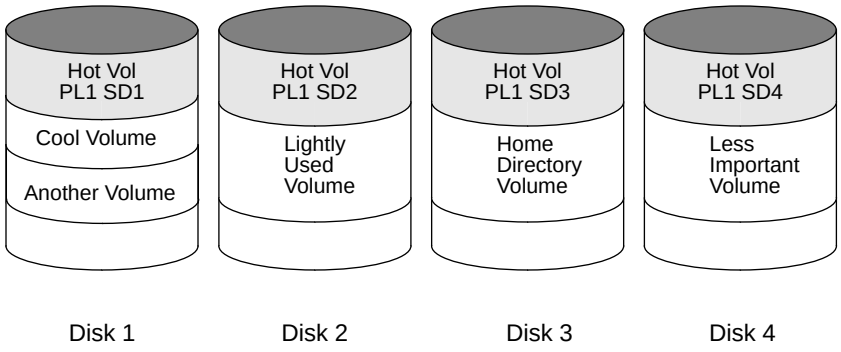


FIGURE 2-1 Use of Striping for Optimal Data Access

2.1.2.1 Striping Guidelines

Follow these guidelines when using striping:

- Never put more than one column of a striped plex on the same physical disk.
- Calculate stripe unit sizes carefully. In general, use a moderate stripe unit size (such as 64K, which is also the default used by `vxassist`). If it is not feasible to set the stripe unit size to the track size, and you do not know the application I/O pattern, use 64 kilobytes for the stripe unit size.

Note – Many modern disk drives have "variable geometry," which means that the track size differs between cylinders (for example, outer disk tracks have more sectors than inner tracks). It is therefore not always appropriate to use the track size as the stripe unit size. For such drives, use a moderate stripe unit size (such as 64K), unless you know the I/O pattern of the application very well.

- Volumes with small stripe unit sizes can often exhibit poor sequential I/O latency if the disks do not have synchronized spindles. Generally, striping over non-spindle-synched disks performs better if used with larger stripe unit sizes and multithreaded, or largely asynchronous, random I/O streams.
- 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 plex of a mirrored volume is striped, be sure to set the policy of the volume to `prefer` for the striped plex. (The default read policy, `select`, does this automatically.)
- If more than one plex of a mirrored volume is striped, make sure the stripe unit size is the same for each striped plex.
- 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 (these are fairly rare).

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

2.1.3 Mirroring

Mirroring is a technique for storing multiple copies of data on a system. When properly applied, mirroring can provide continuous data availability by protecting against data loss due to physical media failure. The use of mirroring improves the chance of data recovery in the event of a system crash or disk failure.

In some cases, mirroring can also improve system performance. Mirroring heavily accessed data not only protects the data from loss due to disk failure, but may also improve I/O 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.

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

- The *round-robin* read policy (`round`), in which read requests to the volume are satisfied in a round-robin manner from all plexes in the volume.
- The *preferred-plex* read policy (`prefer`), in which read requests are satisfied from one specific plex (presumably the plex with the highest performance), unless that plex has failed, in which case another plex is accessed.
- The default read policy (`select`), which selects the appropriate read policy for the configuration (selecting preferred-plex when there is only one striped plex associated with the volume and round-robin in most other cases).

In the configuration example shown in FIGURE 2-2, the read policy of the volume labeled `Hot Vol` should be set to `prefer` for the striped plex labeled `PL1`. In this way, reads going to `PL1` distribute the load across a number of otherwise lightly used disks, as opposed to a single disk.

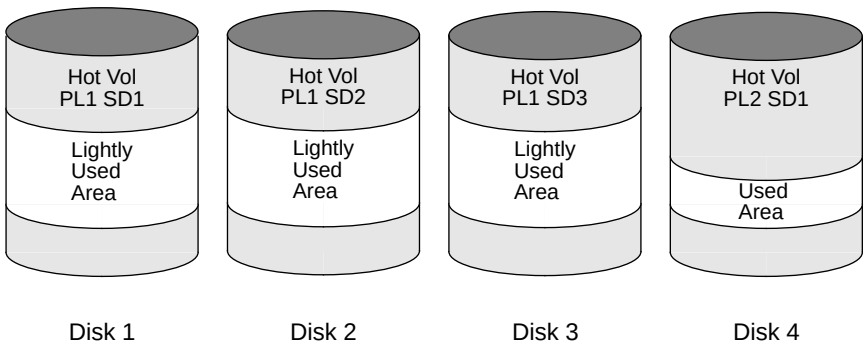


FIGURE 2-2 Use of Mirroring and Striping for Improved Performance

To improve performance for read-intensive workloads, up to 32 plexes 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 plex and across the other half to form another plex; when feasible, this is usually the best way to configure the Volume Manager on a set of disks for best performance with reasonable reliability.

2.1.3.1 Mirroring Guidelines

Follow these guidelines when using mirroring:

- Never place subdisks from different plexes of a mirrored volume on the same physical disk; this action compromises the availability benefits of mirroring and significantly degrades performance. Use of `vxassist` precludes this from happening.
- 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 can frequently be satisfied from this cache, the read/write ratio for physical I/Os through the file system can be significantly biased toward writing when compared to the read/write ratio at the application level.

- Where feasible, use disks attached to different controllers when mirroring or striping. Although most disk controllers support overlapped seeks (enabling seeks to begin on two disks at once), do not configure two plexes 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 (such as those used in most modern workstations and server machines). Mirroring across controllers may, however, be of benefit so that the system can survive a controller failure, in which case the other controller can continue to provide data from the other mirror.
- If one plex exhibits superior performance (due to 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” plex. By default, a volume with one striped plex is configured with preferred reading of the striped plex.

2.1.3.2 Dirty Region Logging Guidelines

Dirty Region Logging (DRL) 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 plex by

maintaining a bitmap and storing this information in a *log subdisk*. Log subdisks are defined for and added to a volume to provide DRL. Log subdisks are independent of plexes; they are ignored as far as the usual plex policies are concerned and are only used to hold the DRL information. Refer to Chapter 1, "Introduction to the Volume Manager" for a complete description of Dirty Region Logging.

Note – Using Dirty Region Logging may impact system performance in a write-intensive environment where there is not much locality of reference.

Follow these guidelines when using DRL:

- In order for Dirty Region Logging to be in effect, the volume must be mirrored.
- At least one log subdisk must exist on the volume for DRL to work. However, only one log subdisk can exist per plex.
- The subdisk that will be used as the log subdisk should not contain any necessary data.
- It is possible to "mirror" log subdisks by having more than one log subdisk (but only one per plex) in the volume. This ensures that logging can continue, even if a disk failure causes one log subdisk to become inaccessible.
- Log subdisks must be configured with 2 or more sectors (preferably an even number, as the last sector in a log subdisk with an odd number of sectors will not be used). The log subdisk size is normally proportional to the volume size. If a volume is less than 2 gigabytes, a log subdisk of 2 sectors is sufficient. The log subdisk size should then increase by 2 sectors for every additional 2 gigabytes of volume size. However, `vxassist` chooses reasonable sizes by default, so you should not normally have to worry about this. In general, use of the default log subdisk length provided by `vxassist` is recommended.
- The log subdisk should not be placed on a heavily used disk, if possible.
- Persistent (non-volatile) storage disks must be used for log subdisks.

2.1.4 Mirroring and Striping

When used together, mirroring and striping provide the advantages of both spreading data across multiple disks and providing redundancy of data.

Mirroring and striping can be used together 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 stripe units in one stripe. Better throughput is achieved because parallel I/O streams can operate concurrently on separate devices.

Since mirroring is most often used to protect against loss of data due to disk 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.

2.1.4.1 Mirroring and Striping Guidelines

Follow these guidelines when using mirroring and striping together:

- Make sure that there are enough disks available for the striped and mirrored configuration. At least two disks are required for the striped plex and one or more *other* disks are needed for the mirror.
- Never place subdisks from one plex on the same physical disk as subdisks from the other plex.
- Follow the striping guidelines described in Section 2.1.2.1, “Striping Guidelines.”
- Follow the mirroring guidelines described in Section 2.1.3.1, “Mirroring Guidelines.”

2.1.5 Using RAID-5

RAID-5 offers many of the advantages of using mirroring and striping together, while requiring less disk space. RAID-5 read performance is similar to that of striping and RAID-5 parity offers redundancy similar to mirroring. Disadvantages of RAID-5 include relatively slow writes.

Note – RAID-5 is not generally seen as a performance improvement mechanism except in cases of high read-to-write ratios shown in the access patterns of the application.

2.1.5.1 RAID-5 Guidelines

In general, the guidelines for mirroring and striping together also apply to RAID-5. In addition, the following guidelines should be observed with RAID-5:

- Only one RAID-5 plex can exist per RAID-5 volume (but there can be multiple log plexes).
- The RAID-5 plex must be derived from at least two subdisks on two or more physical disks. If any log plexes exist, they must belong to disks other than those used for the RAID-5 plex.
- RAID-5 logs can be mirrored and striped.

- If the volume length is not explicitly specified, it will be set to the length of any RAID-5 plex associated with the volume; otherwise, it is set to zero. If the volume length is set explicitly, it must be a multiple of the stripe unit size of the associated RAID-5 plex, if any.
- If the log length is not explicitly specified, it will be set to the length of the smallest RAID-5 log plex that is associated, if any. If no RAID-5 log plexes are associated, it is set to zero.
- Sparse RAID-5 log plexes are not valid.

2.1.6 Hot-Relocation

Hot-relocation provides the advantages of automatically detecting a failure, informing the system administrator of the failure, and attempting to relocate and recover the affected redundant VxVM objects. Refer to Chapter 1, “Introduction to the Volume Manager” for a description of hot-relocation.

2.1.6.1 Hot-Relocation Guidelines

Follow these general guidelines when using hot-relocation:

- The hot-relocation feature is enabled by default. Although it is possible to disable hot-relocation, you should leave it on.
- Although hot-relocation does not require you to designate disks as spares, you should designate at least one disk as a spare within each disk group; this gives you some control over which disks are used for relocation. If no spares exist, VxVM will use any available free space within the disk group. When free space is used for relocation purposes, it is more likely that there may be performance degradation after the relocation.

After hot-relocation occurs, it is also a good idea to designate one or more additional disks as spares to augment the spare space (since some of the original spare space is probably occupied by relocated subdisks).

- If a given disk group spans multiple controllers and has more than one spare disk, it is generally a good idea to set up the spare disks on different controllers (in case one of the controllers fails).
- For hot-relocation to succeed for a mirrored volume, the disk group must have at least one disk that does not already contain one of the volume’s mirrors. This disk should either be a spare disk with some available space or a regular disk with some free space.

- For hot-relocation to succeed for a mirrored and striped volume, the disk group must have at least one disk that does not already contain one of the volume's mirrors or another subdisk in the striped plex. This disk should either be a spare disk with some available space or a regular disk with some free space.
- For hot-relocation to succeed for a RAID-5 volume, the disk group must have at least one disk that does not already contain the volume's RAID-5 plex or one of its log plexes. This disk should either be a spare disk with some available space or a regular disk with some free space.
- If a mirrored volume has a DRL log subdisk as part of its data plex, that plex cannot be relocated. It is therefore advisable to place log subdisks in plexes that contain no data (log plexes).
- Hot-relocation does not guarantee that it will preserve the original performance characteristics or data layout. You should therefore examine the location(s) of the newly-relocated subdisk(s) and determine whether they should be relocated to more suitable disks to regain the original performance benefits.
- Hot-relocation is capable of creating a new mirror of the root disk if the root disk is mirrored and it fails. The `rootdg` disk group should therefore contain sufficient contiguous spare or free space to accommodate the volumes on the root disk (`rootvol` and `swapvol` require contiguous disk space).
- Although it is possible to build VxVM objects on spare disks (using `vxmake` or the Visual Administrator Advanced-Ops menu), it is preferable to use spare disks for hot-relocation only.

2.2 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.

2.2.1 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 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.

Best performance can generally be achieved by striping and mirroring all volumes across a reasonable number of disks, mirroring between controllers when possible. This tends to even out the load between all disks. However, this usually makes the Volume Manager more difficult to administer. If you have a large number of disks (hundreds or thousands), it may make sense to place disks in groups of 10 (using disk groups), where each group is used to stripe and mirror some set of volumes. This still provides good performance and eases the task of administration.

2.2.2 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.

2.2.2.1 Obtaining I/O Statistics

The `vxstat` utility provides access to information for activity on volumes, plexes, 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 average operation time (which reflects the total time through the VxVM interface and is not suitable for comparison against other statistics programs)

VxVM records the preceding three pieces of information for logical I/Os, including reads, writes, atomic copies, verified reads, verified writes, plex reads, and plex writes for each volume. As a result, one write to a two-plex volume results in at least

five operations: one for each plex, 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 plex, and one for the volume.

VxVM also maintains other statistical data. For each plex, 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 `vxstat -r` command 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 afterward.

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
```

Additional volume statistics are available for RAID-5 configurations. See the `vxstat`(1M) manual page for more information.

2.2.2.2 Tracing I/O

The `vxtrace` command is used to trace operations on volumes. `vxtrace` either prints kernel I/O error or I/O trace records to the standard output or writes the records to a file in binary format. Tracing can be applied to specific kernel I/O objects types or to specified objects or devices. For additional information, refer to the `vxtrace`(1M) manual page.

2.2.3 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.

2.2.3.1 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. 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
```

This output helps to identify volumes with an unusually large number of operations or excessive read or write times.

To display disk statistics, type `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
```

At times, it may make sense to move volumes from one disk to another. To move the volume `archive` onto another disk, first 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`, type:

```
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 example, use the following command to move the volume from `disk03` to `disk04`:

```
vxassist move archive !disk03 disk04
```

This command indicates that the volume should be reorganized so that no part remains on `disk03`.

Note – The Visual Administrator provides an easy way to move pieces of volumes between disks and may be preferable to the command-line approach.

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). If done online, converting a volume to use striping requires sufficient free space to store an extra copy of the volume. If sufficient free space is not available, a backup copy can be made instead.

To convert to striping, create a striped plex of the volume and then remove the old plex. For example, to stripe the volume `archive` across disks `disk02`, `disk03`, and `disk04`, use:

```
vxassist mirror archive layout=stripe disk02 disk03 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.

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 need to be restored. Use mirroring or RAID-5 to reduce the chance that a single disk failure will result in failure of a large number of volumes.

Note that 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.

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 high ratio of reads to writes, it is likely that mirroring can significantly improve performance of that volume.

2.2.3.2 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. This helps to explicitly identify the location and size of a hot spot, as well as identifying which application is causing it.

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.

2.3 Tuning the Volume Manager

This section describes the mechanisms for controlling the resources used by the Volume Manager. Adjustments may be required for some of the tunable values to obtain best performance (depending on the type of system resources available).

2.3.1 General Tuning Guidelines

The Volume Manager is tuned for most configurations ranging from small systems to larger servers. In cases where tuning can be used to increase performance on larger systems at the expense of a valuable resource (such as memory), the Volume Manager is generally tuned to run on the smallest supported configuration.

Caution – These tuning changes should be performed with care, as they may adversely affect overall system performance or may even leave the Volume Manager unusable.

Various mechanisms exist for tuning the Volume Manager. Several parameters can be tuned using the global tunable file `/etc/system`. Other values can only be tuned using the command line interface to the Volume Manager.

2.3.2 Tunables

On Solaris, the tunables can be modified by adding lines to the `/etc/system` file and by then rebooting the system. Changed tunables will then be in effect.

For example, to change the default value of a tunable called `vol_tuneme` to a value of 5000, add the following line into the appropriate section of the `/etc/system` file:

```
set vxio:vol_tuneme=5000
```

In many cases, the tunables are contained in the `volinfo` structure, as described in the `vxio`(7) manual page.

The sections that follow describe specific tunables.

2.3.2.1 vol_maxvol

This value controls the maximum number of volumes that can be created on the system. This value can be set to between 1 and the maximum number of minor numbers representable in the system.

The default value for this tunable is half the value of the maximum minor number value on the system.

2.3.2.2 vol_subdisk_num

This tunable is used to control the maximum number of subdisks that can be attached to a single plex. There is no theoretical limit to this number, but for practical purposes it has been limited to a default value of 4096. This default can be changed if required.

2.3.2.3 vol_maxioctl

This value controls the maximum size of data that can be passed into the Volume Manager via an `ioctl` call. Increasing this limit will enable larger operations to be performed. Decreasing the limit is not generally recommended since some utilities depend upon performing operations of a certain size and may fail unexpectedly if they issue oversized `ioctl` requests.

The default value for this tunable is 32768 bytes (32K).

2.3.2.4 vol_maxspecialio

This tunable controls the maximum size of an I/O that can be issued by an `ioctl` call. The `ioctl` request itself may be small, but may have requested a large I/O to be performed. This tunable limits the size of these I/Os. If necessary, a request that exceeds this value may be failed, or the I/O may be broken up and performed synchronously.

The default value for this tunable is 512 sectors (256K).

2.3.2.5 vol_maxio

This value controls the maximum size of logical I/O operations that can be performed without breaking up the request. Physical I/O requests larger than this value will be broken up and performed synchronously. Physical I/Os are broken up based on the capabilities of the disk device and are unaffected by changes to this maximum logical request limit.

The default value for this tunable is 512 sectors (256K).

Raising this limit can only cause difficulties if the size of an I/O causes the process to take more memory or kernel virtual mapping space than exists and thus deadlock. Raising the limit above 512K could cause this problem and may be inadvisable.

The benefits of raising the limit significantly are also likely to be small since I/Os of 256K take sufficiently long to complete that the effects of performing multiple synchronous 256K operations as part of a larger I/O instead of just performing a single larger I/O may be unnoticeable. Lowering the limit significantly can adversely affect performance.

2.3.2.6 vol_maxkiocount

This tunable controls the maximum number of I/Os that can be performed by the Volume Manager in parallel. Additional I/Os that attempt to use a volume device will be queued until the current activity count drops below this value.

The default value for this tunable is 2048.

Since most process threads can only issue a single I/O at a time, reaching the limit of active I/Os in the kernel would require 2K I/O requests being performed in parallel. Raising this limit seems unlikely to provide much benefit except on the largest of systems.

2.3.2.7 vol_default_iodelay

This value is the count in clock ticks that utilities will pause for between issuing I/Os if the utilities have been directed to throttle down the speed of their issuing I/Os, but have not been given a specific delay time. Utilities performing such operations as resynchronizing mirrors or rebuilding RAID-5 columns will use this value.

The default for this value is 50 ticks.

Increasing this value will result in slower recovery operations and consequently lower system impact while recoveries are being performed.

2.3.2.8 voldrl_min_regionsz

With Dirty Region Logging, the Volume Manager logically divides a volume into a set of consecutive regions. The `voldrl_min_regionsz` tunable specifies the minimum number of sectors for a DRL volume region.

The Volume Manager kernel currently sets the default value for this tunable to 1024 sectors.

Larger region sizes will tend to cause the cache hit-ratio for regions to improve. This will improve the write performance, but it will also prolong the recovery time.

2.3.2.9 voldrl_max_drtregs

This tunable specifies the maximum number of dirty regions that can exist on the system at any time. This is a global value applied to the entire system, regardless of how many active volumes the system has.

The default value for this tunable is 2048.

The tunable `voldrl_max_dtregs` can be used to regulate the worse-case recovery time for the system following a failure. A larger value may result in improved system performance at the expense of recovery time.

2.3.2.10 vol_maxparallelio

This tunable controls the number of I/O operations that the `vxconfigd`(1M) daemon is permitted to request from the kernel in a single `VOL_VOLDIO_READ` per `VOL_VOLDIO_WRITE` `ioctl` call.

The default value for this tunable is 256, and it is unlikely that it is desirable to change this value.

2.3.2.11 vol_mvr_maxround

This value controls the granularity of the round-robin policy for reading from mirrors. A read will be serviced by the same mirror as the last read if its offset is within the number of sectors described by this tunable of the last read.

The default for this value is 512 sectors (256K).

Increasing this value will cause less switches to alternate mirrors for reading. This is desirable if the I/O being performed is largely sequential with a few small seeks between I/Os. Large numbers of randomly distributed volume reads are generally best served by reading from alternate mirrors.

2.3.2.12 voliot_iobuf_limit

This value sets a limit to the size of memory that can be used for storing tracing buffers in the kernel. Tracing buffers are used by the Volume Manager kernel to store the tracing event records. As trace buffers are requested to be stored in the kernel, the memory for them is drawn from this pool.

Increasing this size can allow additional tracing to be performed at the expense of system memory usage. Setting this value to a size greater than can readily be accommodated on the system is inadvisable.

The default value for this tunable is 131072 bytes (128K).

2.3.2.13 voliot_iobuf_max

This value controls the maximum buffer size that can be used for a single trace buffer. Requests of a buffer larger than this size will be silently truncated to this size. A request for a maximal buffer size from the tracing interface will result (subject to limits of usage) in a buffer of this size.

The default size for this buffer is 65536 bytes (64K).

Increasing this buffer can provide for larger traces to be taken without loss for very heavily used volumes. Do not increase this value above the value for the `voliot_iobuf_limit` tunable value.

2.3.2.14 voliot_iobuf_default

This value is the default size for the creation of a tracing buffer in the absence of any other specification of desired kernel buffer size as part of the trace `ioctl`.

The default size of this tunable is 8192 bytes (8K).

If trace data is often being lost due to this buffer size being too small, then this value can be tuned to a more generous amount.

2.3.2.15 voliot_errbuf_default

This tunable contains the default size of the buffer maintained for error tracing events. This buffer is allocated at driver load time and is not adjustable for size while the Volume Manager is running.

The default size for this buffer is 16384 bytes (16K).

Increasing this buffer can provide storage for more error events at the expense of system memory. Decreasing the size of the buffer could lead to a situation where an error cannot be detected via the tracing device. Applications that depend on error tracing to perform some responsive action are dependent on this buffer.

2.3.2.16 voliot_max_open

This value controls the maximum number of tracing channels that can be open simultaneously. Tracing channels are clone entry points into the tracing device driver. Each running `vxtrace` command on the system will consume a single trace channel.

The default number of channels is 32. The allocation of each channel takes up approximately 20 bytes even when not in use.

2.3.2.17 vol_checkpt_default

This tunable controls the interval at which utilities performing recoveries or resynchronization operations will load the current offset into the kernel such that a system failure will not require a full recovery, but can continue from the last reached checkpoint.

The default value of the checkpoint is 20480 sectors (10M).

Increasing this size will reduce the overhead of checkpointing on recovery operations at the expense of additional recovery following a system failure during a recovery.

2.3.2.18 volraid_rsrtransmax

This RAID-5 tunable controls the maximum number of transient reconstruct operations that can be performed in parallel. A transient reconstruct operation is one which occurs on a non-degraded RAID-5 volume and was thus not predicted. By limiting the number of these operations that can occur simultaneously, the possibility of flooding the system with many reconstruct operations at the same time is removed, reducing the risk of causing memory starvation conditions.

The default number of these transient reconstructs that can be performed in parallel is 1.

Increasing this size may improve the initial performance on the system when a failure first occurs and before a detach of a failing object is performed, but can lead to possible memory starvation conditions.

2.3.2.19 voliomem_kvmap_size

This tunable defines the size of a kernel virtual memory region used for mapping I/O memory. This must be at least as large as (and should be larger than) `voliomem_max_memory`. This kernel virtual memory is allocated contiguously from kernelmap.

The default size for this tunable is 5M.

2.3.2.20 voliomem_base_memory

This tunable is the amount of memory allocated when the driver is loaded. It is intended to be large enough to support any possible single Volume Manager I/O, but no larger. More memory will be allocated as needed, but additional memory allocations are not guaranteed. If `voliomem_base_memory` is not large enough for an I/O, and no additional memory can be obtained, then the Volume Manager will hang.

The default size for this tunable is 512K.

2.3.2.21 voliomem_max_memory

This tunable is the maximum amount of memory that will be allocated by the Volume Manager for I/Os. This limit can be no larger than `voliomem_kvmap_size`. Smaller values lower the impact of the Volume Manager on other users of the system (for example, a small value ensures that more memory is available for file caching). Larger values improve Volume Manager throughput, particularly for RAID-5.

The default size for this tunable is 4M.

2.3.2.22 `voliomem_chunk_size`

System memory is allocated to and released from the Volume Manager using this granularity. A larger granularity reduces memory allocation overhead (somewhat) by allowing VxVM to keep hold of a larger amount of memory.

The default size for this tunable is 64K.

2.3.3 Tuning for Large Systems

On smaller systems with less than about a hundred drives, tuning should be unnecessary and the Volume Manager should be capable of adopting reasonable defaults for all configuration parameters. On larger systems, however, there may be configurations that require additional control over the tuning of these parameters, both for capacity and performance reasons.

Generally there are only a few significant decisions to be made when setting up the Volume Manager on a large system. One is to decide on the size of the disk groups and the number of configuration copies to maintain for each disk group. Another is to choose the size of the private region for all the disks in a disk group.

Larger disk groups have the advantage of providing a larger free-space pool for the `vxassist`(1M) command to select from, and also provide for the creation of larger arrays. Smaller disk groups do not, however, require as large a configuration database and so can exist with smaller private regions. Very large disk groups can eventually exhaust the private region size in the disk group with the result that no more configuration objects can be added to that disk group. At that point, the configuration will either have to be split into multiple disk groups, or else the private regions will have to be enlarged; this involves re-initializing each disk in the disk group (and may involve reconfiguring everything and restoring from backup).

A general recommendation for users of disk array subsystems is to create a single disk group for each array such that the disk group can be physically moved as a unit between systems.

2.3.3.1 Changing the Number of Configuration Copies for a Disk Group

The selection of the number of configuration copies for a disk group is based entirely on the trade-off between redundancy and performance. As a general (though non-linear) rule, the fewer configuration copies that exist in a disk group, the quicker the group can be initially accessed, the faster the initial start of `vxconfigd`(1M) can proceed, and the quicker transactions can be performed on the disk group.

Caution – The risk of lower redundancy of the database copies is the loss of the configuration database. Loss of the database results in the loss of all objects in the database and thus all data contained in the disk group.

The default policy for configuration copies in the disk group is to allocate a configuration copy for each controller identified in the disk group, or for each target containing multiple addressable disks on the same target. This is generally quite sufficient from the redundancy perspective, but can lead to large numbers of configuration copies under some circumstances.

If this is the case, try to limit the number of configuration copies to a minimum of 4. The location of the copies will be selected as before, according to maximal controller or target spread.

The mechanism for setting the number of copies for a disk group is to use the `vxdg init` command for a new group setup (see the `vxdg`(1M) manual page for details). Alternatively, an existing group's copies can be changed by use of the `vxedit set` command (see the `vxedit`(1M) manual page for details). For example, to set a disk group called `foodg` to contain 5 copies, type:

```
vxedit set nconfig=5 foodg
```

CHAPTER 3

Disk and Disk Group Administration

This chapter describes the Volume Manager operations for managing disks used by the Volume Manager. It also provides information on disk group operations.

Note – Most Volume Manager commands require superuser privileges.

3.1 Standard Disk Devices

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

The Volume Manager supports up to eight partitions (also called *slices*) on a physical disk with Solaris 2.x. These partitions are named, in order, 0 through 7. Partition 2 is reserved to indicate the entire disk.

When a partition is placed under Volume Manager control, a VM disk is assigned to that partition. A symbolic name (the *disk name* or *disk media name*) such as `disk01` can be established to refer to a VM disk.

A partition is addressed through a physical address (generally referred to as the *device name)* of the form `c#t#d#s#`, which comprises the following elements:

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

An example of a device name is `c0t0d0s2`. By convention, `s2` is used to refer to the standard partitioning method used by VxVM. The physical disk is identified to the Volume Manager as `c#t#d#s#`. For many commands, the suffix `s#` is normally optional, though display commands usually report devices with an `s#` suffix. The Volume Manager `vxdiskadm` and `vxdiskadd` utilities take device names without the `s2` suffix. For example, to specify the second disk attached to the first controller to `vxdiskadd`, use the name `c0t1d0`.

The boot disk (which contains the root file system and is used when booting the system) is often identified to the Volume Manager by the device name `c0t0d0`.

A VM disk is typically composed of two regions:

- *private region* — a small area where configuration information is stored. A disk label and configuration records are stored here.
- *public region* — a larger area that covers the remainder of the disk and is used to store subdisks (and allocate storage space).

Three basic disk types exist with VxVM:

- *sliced* — The public and private regions are on different disk partitions.
- *simple* — The public and private regions are on the same disk partition (with the public area following the private area).
- *nopriv* — There is no private region (only a public region for allocating subdisks).

The Volume Manager initializes each new disk with the fewest number of partitions possible (typically two partitions per physical disk). For disk access names ending in `s2`, the default type is `sliced`.

3.2 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.

A system with the Volume Manager installed will always have the default disk group, `rootdg`. By default, operations are directed to the `rootdg` disk group. The system administrator can create additional disk groups, as necessary. Many systems do not need to use more than one disk group, unless they have a large number of disks. Disks do not need to be added to disk groups until the disks are needed to create VxVM objects. They can be initialized and reserved to be added to disk groups later. However, at least one disk (partition) must be added to `rootdg` in order to perform the VxVM installation procedures.

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).

Having excessively large disk groups may cause the private region to fill. In the case of larger disk groups, disks should be set up with larger private areas to log in. A major portion of a disk's private region consists of space for a disk group configuration database containing configuration records for each VxVM object in that disk group. Since each configuration record takes up 256 bytes (or half a block), the number of configuration records that can be created in a disk group is twice the configuration database copy size (which can be obtained from the output of the `vxdg list` *diskgroupname*) command.

3.3 Disk and Disk Group Utilities

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

- The Visual Administrator graphical user interface
- A set of command-line utilities
- The `vxdiskadm` menu-based interface

Utilities discussed in this chapter include:

- `vxdiskadm` — This is the Volume Manager Support Operations menu interface. 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 questions. Default answers are provided for many questions so that common answers can be selected easily.

This chapter describes uses for the `vxdiskadm` utility, but does not describe its use in detail (refer to the *Sun StorEdge Volume Manager 2.6 User's Guide* for information on how to use `vxdiskadm`).

- `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.

- `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) manual 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) manual 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.

3.4 Using Disks

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

3.4.1 Initializing and Adding Disks

There are two levels of initialization for disks in the Volume Manager:

1. Formatting of the disk media itself. This must be done outside of the Volume Manager.
2. Storing identification and configuration information on the disk for use by the Volume Manager. Volume Manager interfaces are provided to step through this level 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.

3.4.1.1 Formatting the Disk Media

To perform the first initialization phase, use the interactive `format` command to do a media format of any disk.

Note – SCSI disks are usually preformatted. The `format` command typically is needed only if the format becomes severely damaged.

3.4.1.2 VxVM Disk Installation

To perform the disk initialization phase, use either the `vxdiskadm` menus or `vxdiskadd`. This section describes how to use `vxdiskadd`. For information on how to use `vxdiskadm` to initialize a single disk or all disks on a controller, refer to Chapter 13, "Menu Interface Operations" of the *Sun StorEdge Volume Manager 2.6 User's Guide*.

You can use `vxdiskadd` to initialize a specific disk. 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 based on what it finds. `vxdiskadd` will check for disks that can be encapsulated (described in Section 3.7.1, "Using vxdisk for Special Encapsulations"), 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; `vxdiskadd` displays a success message when the initialization completes.

At the following prompt, type `y` (or press Return) to continue:

```
Add or initialize disks
Menu: VolumeManager/Disk/AddDisks

 Here is the disk selected.  Output format: [Device_Name]

  c0t1d0

Continue operation? [y,n,q,?] (default: y) y
```

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, a
  new disk group, or 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 free as a replacement disk (not yet added to any disk group), type `none`. After this, you will be asked to select a name for the disk in the disk group:

```
Use a default disk name for the disk? [y,n,q,?] (default: y) y
```

Normally, you should accept the default disk name (unless you prefer to enter a special disk name).

At the following prompt, type `n` to indicate that this disk should not be used as a hot-relocation spare:

```
Add disk as a spare disk for rootdg? [y,n,q,?] (default: n) n
```

When the following screen is displayed, press Return to continue with the operation:

```
  The selected disks will be added to the disk group rootdg with
  default disk names.

  c0t1d0

Continue with operation? [y,n,q,?] (default: y) y
```

If you are certain that there is no data on this disk that needs to be preserved, type `n` at the following prompt:

```
  The following disk device has a valid VTOC, but does not appear to have
  been initialized for the Volume Manager.  If there is data on the disk
  that should NOT be destroyed you should encapsulate the existing disk
  partitions as volumes instead of adding the disk as a new disk.
  Output format: [Device_Name]

  c0t1d0

Encapsulate this device? [y,n,q,?] (default: y)
```

When `vxdiskadm` asks if you want to initialize the disk instead, type `y`:

```
  Instead of encapsulating, initialize? [y,n,q,?] (default: n) y
```

Messages similar to the following should now confirm that disk `c1t0d1` is being placed under Volume Manager control.

```
  Initializing device c0t1d0.

  Adding disk device c0t1d0 to disk group rootdg with disk
  name disk33.
```

3.4.2 Removing Disks

A disk that contains no subdisks can be removed from its disk group with the command:

vxdg [**-g** *groupname*] **rmdisk** *diskname*

where the disk group name need only be specified for a disk group other than the default, `rootdg`.

For example, to remove `disk02` from `rootdg`, 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 the `-k` option to `vxdg` to remove device assignment. Using the `-k` option enables you to remove the disk in spite of the presence of subdisks. For more information, see the `vxdg`(1M) manual page.

Once the disk has been removed from its disk group, you can (optionally) remove it from Volume Manager control completely, by typing:

```
vxdisk rm devicename
```

For example, to remove `c1t0d0` from Volume Manager control, type:

```
vxdisk rm c1t0d0s2
```

In some cases, you may want to remove a disk on which some subdisks are defined. 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 is 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 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.
- Plexes or striped subdisks cannot be allocated on different disks from existing plexes or striped subdisks in the volume.

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

3.4.3 Moving Disks

If you 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
```

Note – This procedure does not save the configurations or data on the disks.

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`).

3.5 Detecting and Replacing Failed Disks

This section discusses how to detect disk failures and replace failed disks. It begins with a discussion of the Volume Manager's hot-relocation feature, which automatically attempts to restore redundant VxVM objects when a failure occurs.

3.5.1 Hot-Relocation

The hot-relocation feature enables a system to automatically react to I/O failures on redundant (mirrored or RAID-5) VxVM objects and restore redundancy and access to those objects. The Volume Manager detects I/O failures on VxVM objects and relocates the affected subdisks to disks designated as spare disks and/or free space within the disk group. VxVM then reconstructs the VxVM objects that existed before the failure and makes them redundant and accessible again. Refer to Chapter 1, "Introduction to the Volume Manager," for a more complete description of hot-relocation.

Note – Hot-relocation is only performed for redundant (mirrored or RAID-5) subdisks on a failed disk. Non-redundant subdisks on a failed disk are not relocated, but the system administrator is notified of their failure.

Hot-relocation is enabled by default and goes into effect without system administrator intervention when a failure occurs. The hot-relocation daemon, `vxrelocd`, detects and reacts to Volume Manager events that signify the following types of failures:

- Disk failure — this is normally detected as a result of an I/O failure from a VxVM object. VxVM attempts to correct the error. If the error cannot be corrected, VxVM tries to access configuration information in the disk's private region. If it cannot access the disk's private region, it considers the disk failed.
- Plex failure — this is normally detected as a result of an uncorrectable I/O error in the plex (which affects subdisks within the plex). For mirrored volumes, the plex is detached.
- RAID-5 subdisk failure — this is normally detected as a result of an uncorrectable I/O error. The subdisk is detached.

When such a failure is detected, `vxrelocd` informs the system administrator of the failure and which VxVM objects are affected (via electronic mail). `vxrelocd` then determines which subdisks (if any) can be relocated. If relocation is possible, `vxrelocd` finds suitable relocation space and relocates the subdisks. Hot-relocation space is chosen from the disks reserved for hot-relocation in the disk group where the failure occurred; if no spare disks are available or additional space is needed, free space in the same disk group is used. Once the subdisks are relocated, each relocated subdisk is reattached to its plex. Finally, `vxrelocd` initiates any appropriate recovery procedures (such as mirror resynchronization for mirrored volumes or data recovery for RAID-5 volumes). The system administrator is notified of the hot-relocation and recovery actions taken.

If relocation is not possible, the system administrator is notified and no further action is taken. Relocation is not possible in the following cases:

- If subdisks are not redundant (that is., they do not belong to mirrored or RAID-5 volumes), they cannot be relocated.
- If there is not enough space available (from spare disks and free space) in the disk group, failing subdisks cannot be relocated.
- If the only available space is on a disk that already contains a mirror of the failing plex, the subdisks in that plex cannot be relocated.
- If the only available space is on a disk that already contains the RAID-5 plex's log plex or one of its healthy subdisks, the failing subdisk in the RAID-5 plex cannot be relocated.
- If a mirrored volume has a Dirty Region Logging log subdisk as part of its data plex, subdisks belonging to that plex cannot be relocated.

- If a RAID-5 volume's log plex or a mirrored volume's DRL log plex fails, a new log plex is created elsewhere (so the log plex is not actually relocated).

It is a good idea to prepare for hot-relocation by designating one or more disks per disk group as hot-relocation spares. For information on how to designate a disk as a spare, see the *Sun StorEdge Volume Manager 2.6 User's Guide*. If no spares are available at the time of a failure or if there is not enough space on the spares, free space will automatically be used. By designating spare disks, you have some control over which space is used for relocation purposes in the event of a failure. If the combined free space and space on spare disks is not sufficient or does not meet the redundancy constraints, the subdisks are not relocated.

After a successful relocation occurs, you may need to remove and replace the failed disk (see Section 3.5.3, "Replacing Disks"). Depending on the locations of the relocated subdisks, you may choose to move the relocated subdisks elsewhere after hot-relocation occurs (see Section 3.5.4, "Moving Relocated Subdisks").

3.5.1.1 Modifying `vxrelocd`

As long as `vxrelocd` is running, hot-relocation is turned on. You should leave hot-relocation turned on so that you can take advantage of this feature if a failure occurs. However, if you choose to disable this feature for some reason (possibly because you do not want the free space on some of your disks used for relocation), you can do so by preventing `vxrelocd` from starting at system startup time. Refer to the *Sun StorEdge Volume Manager 2.6 Installation Guide* for information on how to disable hot-relocation at system startup. Alternatively, you can stop hot-relocation at any time by killing the `vxrelocd` process (this should not be done while a hot-relocation attempt is in progress).

You can make some minor changes to the way `vxrelocd` behaves by either editing the `vxrelocd` line in the startup file that invokes `vxrelocd` (`/etc/rc2.d/S95vxvm-recover`) or killing the existing `vxrelocd` process and restarting it with different options. After making changes to the way `vxrelocd` is invoked in the startup file, you need to reboot the system so that the changes will go into effect. If you choose to kill and restart the daemon instead, make sure that hot-relocation is not in progress when you kill the `vxrelocd` process. You should also restart the daemon immediately so that hot-relocation can take effect if a failure occurs.

You can alter `vxrelocd`'s behavior in the following ways:

- By default, `vxrelocd` sends electronic mail to `root` when failures are detected and relocation actions are performed. You can instruct `vxrelocd` to notify additional users by adding the appropriate user names and invoking `vxrelocd` as follows:

```
vxrelocd root user_name1 user_name2 &
```

- To reduce the impact of recovery on system performance, you can instruct `vxrelocd` to increase the delay between the recovery of each region of the volume, as follows:

```
vxrelocd -o slow[=IOdelay] root &
```

where the optional *IOdelay* indicates the desired delay (in milliseconds). The default value for the delay is 250 milliseconds.

3.5.1.2 Displaying Spare Disk Information

The command `vxdg spare` can be used to display information about all of the spare disks available for relocation. The output of this command lists the following type of information:

```
GROUP        DISK         DEVICE       TAG          OFFSET    LENGTH    FLAGS
rootdg       disk02       c0t2d0s2     c0t2d0       0         658007    s
```

In this case, `disk02` is the only disk designated as a spare. The `LENGTH` field indicates how much spare space is currently available on this disk for relocation.

The following commands can also be used to display information about disks that are currently designated as spares:

- `vxdisk list` — lists disk information and displays spare disks with a `spare` flag.
- `vxprint` — lists disk and other information and displays spare disks with a `SPARE` flag.

3.5.2 Detecting Failed Disks

Note – VxVM's hot-relocation feature automatically detects disk failures and notifies the system administrator of the failures via electronic mail.
If hot-relocation is disabled or you miss the electronic mail, you may notice disk failures through the output of the `vxprint` command or by using the Visual Administrator to look at the status of the disks. You may also see driver error messages on the console or in the system messages file.

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

3.5.2.1 Partial Disk Failure

If hot-relocation is enabled when a plex or disk is detached by a failure, mail indicating the failed objects is sent to `root`. If a partial disk failure occurs, the mail should identify the failed plexes. For example, if a disk containing mirrored volumes experiences a failure, you might receive a mail message containing information such as:

```
To: root
Subject: Volume Manager failures on host teal

Failures have been detected by the Volume Manager:

failed plexes:
  home-02
  src-02
```

See Section 3.5.1.1, “Modifying vxrelocd,” for information on how to have the mail sent to users other than `root`.

You can determine which disk is causing the failures in the above message by running the command:

```
vxstat -s -ff home-02 src-02
```

This produces 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` (and that subdisks `disk02-03` and `disk02-04` are affected).

Hot-relocation should automatically relocate the affected subdisks and initiate any necessary recovery procedures. However, if relocation is not possible or the hot-relocation feature is disabled, you may have to investigate the problem and attempt to recover the plexes. Sometimes these errors are caused by cabling failures, so you should check the cables connecting your disks to your system. If there are any obvious problems, correct them and recover the plexes with the command:

```
vxrecover -b home src
```

This command will start a recovery of the failed plexes in the background (the command will return before the operation is done). If an error message is displayed later, or if the plexes become detached again and there are no obvious cabling failures, the disk should be replaced (see Section 3.5.3, “Replacing Disks”).

3.5.2.2 Complete Disk Failure

If a disk fails completely and hot-relocation is enabled, the mail message will list the disk that has failed and all plexes that use the disk. For example, you may receive mail containing information similar to this:

```
To: root
Subject: Volume Manager failures on host teal

Failures have been detected by the Volume Manager:

failed disks:
  disk02

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

failing disks:
  disk02
```

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

Again, the problem may be a cabling error. If the problem is not a cabling error, the disk should be replaced (see Section 3.5.3, “Replacing Disks”).

3.5.3 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.

Note – Do not choose the old disk drive as a replacement even though 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, 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, type:

```
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
rootvol      root      Started
swapvol      swap      Started
```

To restart volume `mkting` so that it can be restored from backup, type:

```
vxvol -o bg -f start mkting
```

The `-o bg` option combination causes any plexes to be resynchronized as 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:

1. Detaching the disk from its disk group.

2. 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:

     home 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 are likely to be disabled, you should quit from `vxdiskadm` and save the volume. Either back up the volume or move the volume off of the disk. To move the volume `mkting` to a disk other than `disk02`, type:

```
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, specify a replacement disk by selecting item `5 (Replace a failed or removed disk)` from the `vxdiskadm` main menu.

3.5.4 Moving Relocated Subdisks

This section describes how to move subdisks after hot-relocation occurs. When hot-relocation occurs, subdisks are relocated to spare disks and/or available free space within the disk group. The new subdisk locations may not provide the same performance or data layout that existed before hot-relocation took place. You may therefore want to move the relocated subdisks (after hot-relocation is complete) to improve performance. Alternatively, you may want to move the relocated subdisks off the spare disk(s) to keep the spare disk space free for future hot-relocation needs. Another reason for moving subdisks is to re-create the configuration that existed before hot-relocation occurred.

During hot-relocation, one of the electronic mail messages sent to `root` should be similar to the following:

```
To: root
Subject: Volume Manager failures on host teal

Attempting to relocate subdisk disk02-03 from plex home-02.
Dev_offset 0 length 1164 dm_name disk02 da_name c0t5d0s2.
The available plex home-01 will be used to recover the data.
```

This message provides information about the characteristics of the subdisk before relocation and can be used to decide where to move the subdisk after relocation. In addition, a message similar to the following should indicate the new location for the relocated subdisk:

```
To: root
Subject: Attempting VxVM relocation on host teal

Volume home Subdisk disk02-03 relocated to disk05-01,
but not yet recovered.
```

Before moving any relocated subdisks, you should fix or replace the disk that experienced the failure (as described in previous sections). Once this is done, you can move a relocated subdisk back to the original disk. For example, you can move the relocated subdisk `disk05-01` back to `disk02` as follows:

```
vxassist -g rootdg move home !disk05 disk02
```

Note – During subdisk move operations, RAID-5 volumes are not redundant.

3.6 Working With Disk Groups

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

3.6.1 Creating a Disk Group

A new disk group can be created on a physical disk using `vxdiskadd`. If you are using `vxdiskadd`, specify a new disk group name when prompted for a disk group. Disk groups can also be created using the operation `vxdg init`. To create a disk group with the `vxdg` utility, type:

```
vxdg init diskgroup diskname=devicename
```

For example, to create a disk group named `mktdg` on device `c1t0d0s2`, type:

```
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.

3.6.2 Using Disk Groups

Most Volume Manager commands enable 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 (block) 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 enables 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.

3.6.3 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 diskgroup
```

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.

3.6.4 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 specify the configuration again.

To move a disk group between systems:

1. **On the first system, stop all volumes in the disk group, then deport (disable local access to) the disk group by typing:**

```
vxdg deport diskgroup
```

2. **Move all the disks to the second system and import (enable local access to) the disk group on the second system by typing:**

```
vxdg import diskgroup
```

3. **Perform the necessary steps (system-dependent) to make the second system and VxVM recognize the new disks.**

   This may require a reboot, in which case the `vxconfigd` daemon will be restarted and will also recognize the new disks. If you do not reboot, use the command `vxdctl enable` to restart `vxconfigd` so that VxVM also recognizes the disks.

4. **After the disk group is imported, start all volumes in the disk group by typing:**

```
vxrecover -g diskgroup -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.

Note – 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 and its data 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, type:

```
vxdisk clearimport devicename ...
```

Note – It is possible to clear the locks during import by typing:
`vxdg -C import` *diskgroup*

Note – Be careful when using the `vxdisk clearimport` or `vxdg -C import` command on systems that really do have dual-ported disks, as clearing the locks enables those disks to be accessed simultaneously and could result in corrupted data.

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 group has no valid
configuration copies
```

This message indicates a fatal situation, which requires hardware repair or the creation of a new disk group.

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

This message indicates a recoverable situation.

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

```
vxdg -f import diskgroup
```

Note – Be careful when using the `-f` option here, as it can cause the same disk group to be imported twice from disjoint sets of disks, causing the disk group to become inconsistent.

These operations can be performed using `vxdiskadm`. To deport a disk group via `vxdiskadm`, select menu 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.

3.6.4.1 Renaming Disk Groups

Only one disk group of a given name can exist per system. It is therefore not possible to import or deport a disk group when the target system already has a disk group of the same name. To circumvent this problem, the Volume Manager enables you to temporarily or permanently rename a disk group during import or deport.

For instance, since every system running the Volume Manager must have a single `rootdg` default disk group, importing or deporting `rootdg` across systems presents a problem in that there cannot be two `rootdg` disk groups on the same system. This problem can be overcome by renaming the `rootdg` disk group during the import or deport.

To give a new name to the disk group during import, type:

```
vxdg [-t] -n newdg_name import diskgroup
```

If the `-t` option is included, the import is temporary and will not persist across reboots. In this case, the stored name of the disk group remains unchanged on its original host, but the disk group will be known as *newdg_name* to the importing host. If the `-t` option is not used, the name change will be permanent.

As with importing, to rename a disk group on deport, type:

```
vxdg [-h hostname] -n newdg_name deport diskgroup
```

When renaming on deport, you can use the `-h` *hostname* option to assign a lock to an alternate host. This ensures that the disk group is automatically imported when the alternate host reboots.

The following set of steps can be used to temporarily move the `rootdg` disk group from one host to another (for repair work on the root volume, for instance) and then move it back:

1. **On the original host, identify the disk group ID of the** `rootdg` **disk group to be imported to the other host**:

```
vxdisk -s list
```

This command results in output that includes disk group information similar to the following.

```
dgname: rootdg
dgid:   774226267.1025.tweety
```

2. **On the importing host, import and rename the** `rootdg` **disk group by typing**:

```
vxdg -tC -n newdg_name import diskgroup
```

where `-t` indicates a temporary import name; `-C` clears import locks; `-n` specifies a temporary name for the `rootdg` to be imported (so that it does not conflict with the existing `rootdg`); and *diskgroup* is the disk group ID of the disk group being imported (`774226267.1025.tweety`, for example).

If a reboot or crash occurs at this point, the temporarily imported disk group will become unimported and will require a reimport.

3. **After the necessary work has been done on the imported** `rootdg`**, deport it back to its original host by typing**:

```
vxdg -h hostname deport diskgroup
```

where *hostname* is the name of the system whose `rootdg` is being returned (the system's name can be confirmed with the command `uname -n`).

This command removes the imported `rootdg` from the importing host and returns locks to its original host. The original host will then autoimport its `rootdg` on its next reboot.

3.6.4.2 Reserving Minor Numbers for Disk Groups

Since disk groups can be moved between systems, volume device numbers should be allocated in separate ranges for each disk group so that all disk groups in a group of machines can be moved around without causing device number collisions.

VxVM enables the administrator to select a range of minor numbers for a specified disk group. This range is used during the creation of a volume, guaranteeing that each volume will have the same minor number across reboots or reconfigurations. If two disk groups have overlapping ranges, an import collision will be detected and some avoidance or renumbering mechanism will then be needed.

You can set a base volume device minor number for a disk group with the command:

```
vxdg init diskgroup minor=base_minor devicename
```

Volume device numbers for a disk group will be chosen to have minor numbers starting at this *base_minor* number. Minor numbers (in Solaris) can range up to 131071. A reasonably sized range should be left at the end for temporary device number remappings (in the event that two device numbers still conflict).

If no `minor` operand is specified on the `vxdg init` command line, the Volume Manager chooses a random number of at least 1000 that is a multiple of 1000, and yields a usable range of 1000 device numbers. This default number is chosen so that it does not overlap within a range of 1000 of any currently imported disk groups, and does not overlap any currently allocated volume device numbers.

Note – The default policy is likely to ensure that a small number of disk groups can be merged successfully between a set of machines. However, in cases where disk groups will be merged automatically using fail-over mechanisms, the administrator should select ranges that are known to avoid overlap.

For further information on minor number reservation, refer to the `vxdg`(1M) manual page.

3.7 Using Special Devices

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

3.7.1 Using vxdisk for Special Encapsulations

Encapsulation is a process that converts existing partitions on a specified disk to volumes. If any partitions contain file systems, /etc/vfstab entries are modified so that the file systems are mounted on volumes instead.

The normal disk encapsulation procedure requires that some free space be available on the disk for storing Volume Manager identification and configuration information. This free space cannot be included in any other partitions. (Refer to the *Sun StorEdge Volume Manager 2.6 Installation Guide* and the vxencap(1M) manual page for further information.)

In some cases, you may want to encapsulate a disk that does not have any space that can be used for the Volume Manager private region partition. 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 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, by typing:

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

To create volumes for other partitions on the disk drive, add the device to a disk group, determine 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 the disk. When space is made available on 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.

3.7.2 Using `vxdisk` for RAM Disks

Note – This section only applies to systems with 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 that 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, plexes made from volatile devices are always recovered by copying data from non-volatile plexes.

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

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

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

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

```
vxvol set startopts=norecov volume_name
```

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.

3.7.3 Using `vxdisk` For Displaying Multipathing Information

In Volume Manager, all the physical disks connected to the system are represented as metadevices with one or more physical access paths depending on whether the disk is a single disk or part of a multiported disk array connected to the system. The `vxdisk` utility is used to display the various paths of a metadevice, and also to display the status of each path such as *failed* or *active*.

For example, to display details on a particular disk, type:

```
vxdisk list disk01
```

The Volume Manager returns the following display:

```
Device    c2t0d0s2
devicetag c2t0d0
type      sliced
hostid    aparajita
disk      name=disk01 id=861086917.1052.aparajita
group     name=rootdg id=861086912.1025.aparajita
flags     online ready autoconfig autoimport imported
pubpaths  block=/dev/vx/dmp/c2t0d0s4 char=/dev/vx/rdmp/c2t0d0s4
privpaths block=/dev/vx/dmp/c2t0d0s3 char=/dev/vx/rdmp/c2t0d0s3
version   2.1
iosize    min=512 (bytes) max=2048 (blocks)
public    slice=4 offset=0 len=1043840
private   slice=3 offset=1 len=1119
update    time=861801175 seqno=0.48
headers   0 248
configs   count=1 len=795
logs      count=1 len=120
Defined regions
config   priv 000017-000247[000231]:copy=01 offset=000000 enabled
config   priv 000249-000812[000564]:copy=01 offset=000231 enabled
log     priv 000813-000932[000120]:copy=01 offset=000000 enabled
Multipathing information:
numpaths:       2
c2t0d0s2        active
c1t0d0s2        failed
```

CHAPTER 4

Volume Administration

This chapter provides information on how to maintain a system configuration under VxVM control. It includes information about how to create, remove, and maintain VxVM objects (volumes, plexes, and subdisks). Information is presented in the form of both overviews of relevant utilities and more detailed descriptions on how to perform particular operations using the utilities. Online backup information is also included.

Note – Most Volume Manager commands require superuser privileges.

4.1 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, plexes, and volumes. Volumes are composed of two types of objects — plexes and subdisks. A plex 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.

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:

- Automatically — Volumes can be created and manipulated 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 plexes and subdisks. `vxassist` can also be used to modify existing volumes, in which case it also automatically modifies any underlying or associated objects. `vxassist` uses default values for many volume attributes, unless specific values are provided.

- Manually — Volumes can be built in a “bottom-up” style by first creating the volume’s subdisks, then its plexes, then the volume itself. The new volume and its associated plexes and subdisks can then be manipulated, as necessary. This approach often requires detailed user input and a good understanding of VxVM concepts. The manual approach is useful for creating and manipulating volumes with specific, non-default attributes.

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 *Sun StorEdge Volume Manager 2.6 User’s Guide*.

This chapter contains detailed descriptions of Volume Manager operations, along with examples that illustrate the use of VxVM utilities. Many of the concepts and procedures in this chapter are normally “hidden” from the casual user or are performed automatically through use of the higher level commands such as `vxassist`. Some of the commands and procedures documented in this chapter do not normally need to be performed by system administrators.

4.2 VxVM Utility Descriptions

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

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 require a thorough understanding of Volume Manager concepts. `vxassist` is capable of accomplishing alone many 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 not conflict with the existing VxVM utilities or preclude their use. Objects created by `vxassist` are compatible and interoperable with objects created manually using the other VxVM utilities and interfaces, and vice versa.

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 online backup of existing volumes
- Provides an estimate of the maximum size for a new or existing volume

For detailed information about how to use vxassist, refer to the vxassist(1M) manual page.

4.2.1.1 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.

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.

4.2.1.2 How vxassist Works

The vxassist utility enables you 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, `vxassist` 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.

The `vxassist` command typically takes the following form:

```
vxassist keyword volume_name [attributes...]
```

where *keyword* selects the specific operation to perform. The first argument after any `vxassist` keyword is always a volume name. The volume name is followed by a set of attributes. For details on available `vxassist` keywords and attributes, refer to the `vxassist`(1M) manual page.

`vxassist` creates and manipulates volumes based on a set of established defaults, but also enables you to supply preferences for each operation.

4.2.1.3 `vxassist` Defaults

The `vxassist` invocation is designed to be as simple as possible, but it can be customized when necessary. `vxassist` uses a set of tunable parameters, which can be specified in defaults files or at the command line. The tunable parameters default 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 an alternate defaults file.
- Alternate defaults file — A non-standard defaults file, specified with the command `vxassist -d` *alt_defaults_file*.
- Command line — The tunable values specified on the command line override any values specified internally or in defaults files.

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 `vxassist` command line (refer to the `vxassist`(1M) manual page for details).

The following is a sample vxassist defaults file:

```
# by default:
#  create unmirrored, unstriped volumes
#  allow allocations to span drives
#  with RAID-5 create a log, with mirroring don’t create a log
#  align allocations on cylinder boundaries
    layout=nomirror,nostripe,span,nocontig,raid5log,noregionlog,
    diskalign

#  use the fsgen usage type, except when creating RAID-5 volumes
    usetype=fsgen

#  allow only root access to a volume
    mode=u=rw,g=,o=
    user=root
    group=root

#  when mirroring, create two mirrors
    nmirror=2

#  for regular striping, by default create between 2 and 8 stripe
#  columns
    max_nstripe=8
    min_nstripe=2

#  for RAID-5, by default create between 3 and 8 stripe columns
    max_nraid5stripe=8
    min_nraid5stripe=3

#  create 1 log copy for both mirroring and RAID-5 volumes, by
default
    nregionlog=1
    nraid5log=1

#  by default, limit mirroring log lengths to 32Kbytes
    max_regionloglen=32k

#  use 64K as the default stripe unit size for regular volumes
    stripe_stwid=64k

#  use 16K as the default stripe unit size for RAID-5 volumes
    raid5_stwid=16k
```

4.2.2 Manipulating the Volume Configuration Daemon With `vxdctl`

The volume configuration daemon (`vxconfigd`) is the interface between the VxVM utilities and the kernel `vxconfig` device driver. The `vxconfig` 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 `volboot` file, which indicates the locations of some root configuration copies.

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

- Performing administrative tasks related to the state of the `vxconfigd` daemon
- Managing boot information and various aspects of the VxVM root configuration initialization
- Manipulating the contents of the `volboot` file, which contains a list of some disks containing root configuration databases (this is not normally necessary, as VxVM usually locates all disks on the system automatically)
- Reconfiguration of the DMP database that reflects new disk devices attached to the system, and any removal of the disk devices from the system.
- Creating DMP device nodes in the device directory `/dev/vx/dmp`, and `/dev/vx/rdmp`.

For detailed information about how to use `vxdctl`, refer to the `vxdctl`(1M) manual page.

4.2.3 Removing and Modifying Entities With `vxedit`

The `vxedit` utility has two functions:

- `vxedit` enables you to modify certain records in the volume management databases. Only fields that are not volume usage-type-dependent can be modified.
- `vxedit` can be used to remove or rename VxVM objects.

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

- A subdisk that is associated with a plex
- A plex that is associated with a volume

Note – Using the recursive suboption (`-r`) to the removal option of `vxedit` removes all objects from the specified object downward. In this way, a plex and its associated subdisks, or a volume and its associated plexes 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`(1M) manual page.

4.2.4 Creating VxVM Objects With `vxmake`

The `vxmake` utility is used to add a new volume, plex, 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 on the command line, 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 that 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* option specifies a file name. The following is a sample description file:

```
#rectyp #name       #options
sd      disk3-01    disk=disk3 offset=0 len=10000
sd      disk3-02    disk=disk3 offset=25000 len=10480
sd      disk4-01    disk=disk4 offset=0 len=8000
sd      disk4-02    disk=disk4 offset=15000 len=8000
sd      disk4-03    disk=disk4 offset=30000 len=4480
plex    db-01       layout=STRIPE ncolumn=2 stwidth=16k
                sd=disk3-01:0/0,disk3-02:0/10000,disk4-01:1/0,disk4-02:1/8000, \
                     disk4-03:1/16000
sd      ramd1-01    disk=ramd1 len=640
                    comment="Hot spot for dbvol
plex    db-02       sd=ramd1-01:40320
vol     db          usetype=gen plex=db-01,db-02
                    readpol=prefer prefname=db-02
                    comment="Uses mem1 for hot spot in last 5m
```

This description file specifies a volume with two plexes. The first plex has five subdisks on physical disks. The second plex is preferred and has one subdisk on a volatile memory disk.

For detailed information about how to use `vxmake`, refer to the `vxmake`(1M) manual page.

4.2.5 Correcting Plex Problems With `vxmend`

The `vxmend` utility performs miscellaneous VxVM usage-type-specific operations on volumes, plexes, and subdisks. These operations are used to fix simple problems in configuration records (such as clearing utility fields, changing volume or plex states, and offlining or onlining volumes or plexes).

`vxmend` is used primarily to escape from a state that was accidentally reached. The offline and online functions can be performed with disk-related commands.

For detailed information about how to use `vxmend`, refer to the `vxmend`(1M) manual page.

4.2.6 Performing Plex Operations With `vxplex`

The `vxplex` utility performs VxVM operations on plex or on volume-and-plex 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 be used to do the following:

- Attach a plex to a volume or detach a plex from a volume. A detached plex does not participate in I/O activity to the volume, but remains associated with the volume. A detached plex is reattached when a volume is next started.
- Dissociate a plex from the volume with which it is associated. When a plex is dissociated, its relationship to the volume is completely broken. At that point, the plex is available for other uses and can be associated with a different volume. This functionality is useful as part of a backup procedure.
- Copy the contents of the specified volume to the named plex(es). This operation can be used to make a copy of a volume (for backup purposes) without mirroring the volume in advance.
- Move the contents of one plex to a new plex. This is useful for moving a plex on one disk to a different location.

For detailed information about how to use `vxplex`, refer to the `vxplex`(1M) manual page.

4.2.7 Printing Configuration Information With `vxprint`

The `vxprint` utility provides a flexible method of displaying information from records in a 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 between VxVM objects. `vxprint` can tailor its 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`(1M) 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`(1M) 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 for Volume Manager objects 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`(1M) manual page and Chapter 2, “VxVM Performance Monitoring.”

4.2.10 Tracing Volume Operations With `vxtrace`

The `vxtrace` utility prints kernel I/O error or I/O error trace event records on the standard output or writes them to a file in binary format. Binary trace records written to a file can be read back and formatted by `vxtrace` as well.

If no operands are given, then either all error trace data or all I/O trace data on all virtual disk devices are reported. With error trace data, it is possible to select all accumulated error trace data, to wait for new error trace data, or both (the default). Selection can be limited to a specific disk group, to specific VxVM kernel I/O object types, or to particular named objects or devices.

For detailed information about how to use `vxtrace`, refer to the `vxtrace`(1M) manual page and Chapter 2, "VxVM Performance Monitoring."

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 (however, it is seldom useful to stop a volume).

One of these read policies can be selected:

- `round` — prescribes round-robin reads of enabled plexes
- `prefer` — prescribes preferential reads from a specified plex
- `select`— chooses an appropriate policy

For detailed information about how to use `vxvol`, refer to the `vxvol`(1M) manual page.

4.3 Subdisk Operations

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 log subdisks
- Dissociating a subdisk
- Changing a subdisk
- Moving a subdisk
- Splitting a subdisk
- Joining a subdisk

4.3.1 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, type:

```
vxmake sd name disk,offset,len
```

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

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

Commands take sizes in sectors. Adding a suffix (such as `k`, `m`, or `g`) changes the unit of measure.

Note – To preserve (encapsulate) data that exists on the disk, a plex and volume must be created to cover that data.

4.3.2 Removing Subdisks

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

```
vxedit rm subdisk_name
```

For example, to remove a subdisk labeled `disk02-01`, you would type:

```
vxedit rm disk02-01
```

4.3.3 Displaying Subdisk Information

The `vxprint` utility displays information about VxVM objects. To display general information for all subdisks, type:

```
vxprint -st
```

The `-s` option specifies 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, type:

```
vxprint -l subdisk_name
```

For example, to obtain all information on a subdisk labeled `disk02-01`, you would type:

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

This command provides the following information:

```
Disk group: rootdg

Subdisk:  disk02-01
info:     disk=disk02 offset=0 len=205632
assoc:    vol=mvol plex=mvol-02 (offset=0)
flags:    enabled
device:   device=c2t0d1s2 path=/dev/dsk/c2t0d1s4 diskdev=32/68
```

4.3.4 Associating Subdisks

Associating a subdisk with a plex places the amount of disk space defined by the subdisk at a specific offset within the plex. 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 plexes, depending on the overall state of the configuration.

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

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

For example, to create the plex `home-1` and associate subdisks `disk02-01`, `disk02-00`, and `disk02-02` with the plex `home-1` during the plex creation process, you would type:

```
vxmake plex home-1 sd=disk02-01,disk02-00,disk02-02
```

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 plex and then associate each of the subdisks with that plex. In this example, the subdisks are associated to the plex in the order they are listed (after the `sd=`); the disk space defined as `disk02-01` will be first, the disk space of `disk02-00` is second, and `disk02-02` is third.

This method of associating subdisks is convenient during initial configuration. Subdisks can also be associated with a plex that already exists. To associate one or more subdisks with an existing plex, type:

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

For example, to associate subdisks labeled `disk02-01`, `disk02-00`, and `disk02-02` with a plex labeled `home-1`, you would type:

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

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

You would use the `-l` option if you have created a sparse plex for a particular volume, and you want to make this plex complete. To make the plex complete, you must create a subdisk the exact size needed to fill the hole in the sparse plex, and then associate the subdisk with the plex by specifying the offset of the beginning of the hole in the plex. To do this, you would type:

```
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 plex.

For striped subdisks, a column number and column offset for the subdisk can be specified:

```
vxsd -l column_#/offset assoc plex_name sd_name ...
```

If only one number is specified with the `-l` option for striped plexes, the number is interpreted as a column number and the subdisk is associated at the end of the column.

4.3.4.1 Associating Log Subdisks

Log subdisks are subdisks that are defined for and added to a plex that is to become part of a volume using Dirty Region Logging. Dirty Region Logging is enabled for a volume when the volume is mirrored and has at least one log subdisk.

For a description of Dirty Region Logging, refer to Section 1.1.8, “Dirty Region Logging” and Section 2.1.3.2, “Dirty Region Logging Guidelines”. Log subdisks are ignored as far as the usual plex policies are concerned, and are only used to hold the dirty region log.

Note – Only one log subdisk can be associated with a plex. Because this log subdisk is frequently written, make sure you position it on a disk that is not heavily used. Placing a log subdisk on a heavily used disk can degrade system performance.

To add a log subdisk to an existing plex, type:

```
vxsd aslog plex subdisk
```

where *subdisk* is the name of the subdisk to be used as a log subdisk. The plex must be associated with a mirrored volume before DRL will take effect.

For example, to associate a subdisk labeled `disk02-01` with a plex labeled `vol01-02` (which is already associated with volume `vol01`), you would type:

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

Note – You can also use the following command to add a log subdisk to an existing volume:
`vxassist addlog` *volume_name disk*
This command automatically creates a log subdisk within a log plex for the specified volume.

4.3.5 Dissociating Subdisks

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

```
vxsd dis subdisk_name
```

To dissociate a subdisk labeled `disk02-01` from the plex with which it is currently associated, type:

```
vxsd dis disk02-01
```

Note – You can also remove subdisks by typing:
`vxsd -orm dis` *subdisk_name*

4.3.6 Changing Subdisk Information

Use the `vxedit` utility to change information related to subdisks. To change information relating to a subdisk, type:

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

For example, to change the comment field of a subdisk labeled `disk02-01`, type:

```
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 `putil0` field prevents the subdisk from being used as part of a plex, if it is not already.

4.3.7 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 plex, then the data stored on the original subdisk is copied to the new subdisk, the old subdisk is dissociated from the plex, and the new subdisk is associated with the plex, at the same offset within the plex as the source subdisk. To move a subdisk, type:

```
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 plex on an active (ENABLED) volume.
- The new subdisk must not be associated with any other plex.

4.3.8 Splitting Subdisks

Splitting a subdisk divides an existing subdisk into two subdisks. To split a subdisk, type:

```
vxsd -s size split subdisk_name newsd1 newsd2
```

where *subdisk_name* is the name of the original subdisk, *newsd1* 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. The `-s` option is required to specify the size of the *first* of the two subdisks that will be created. The second subdisk will take up the remaining space used by the original subdisk.

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

To split the original subdisk into more than two subdisks, repeat the previous command as many times as necessary on the resulting subdisks.

4.3.9 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 plex, and be contiguous in that plex. To join a subdisk, type:

```
vxsd join subdisk1 subdisk2 new_subdisk
```

4.4 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 creating multiple plexes for a single volume, where each plex contains an identical copy of the volume's data. Since each plex must reside on different disks, the replication provided by mirroring 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
- Performing backups using mirroring
- Associating a plex
- Dissociating and removing a plex
- Listing all plexes
- Displaying plexes
- Changing plex attributes
- Changing plex status
- Moving plexes
- Copying plexes

4.4.1 Creating Plexes

You create a plexes by identifying subdisks and associating them to the plex that you want to create. You use the `vxmake` command to create VxVM objects.

To create a plex from existing subdisks, type:

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

For example, to create a concatenated plex labeled `vol01-02` using two existing subdisks labeled `disk02-01` and `disk02-02` you would type:

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

To create a striped plex, additional attributes need to be specified. For example, to create a striped plex named `pl-01` with a stripe width of 32 sectors and 2 columns you would type:

```
vxmake plex pl-01 layout=STRIPE stwidth=32 ncolumn=2\
sd=disk01-01,disk02-01
```

4.4.2 Performing Backups Using Mirroring

If a volume is mirrored, it can be backed up 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.

Note – The information in this section does not apply to RAID-5.

To perform a backup of a mirrored volume on an active system:

1. **Optionally ask users to stop activity for a short time to improve the consistency of the backup.**
2. **Dissociate one of the volume's mirrors (`vol01-01` is used in this example):**

```
vxplex dis vol01-01
```

3. **Create a new, temporary volume using the dissociated plex:**

```
vxmake -U gen voltemp plex=vol01-01
```

4. **Start the temporary volume:**

```
vxvol start tempvol
```

5. **Perform appropriate backup procedures using the temporary volume.**
6. **Stop the temporary volume:**

```
vxvol stop tempvol
```

7. **Dissociate the backup plex from its temporary volume:**

```
vxplex dis vol01-01
```

8. **Reassociate the backup plex with its original volume to regain redundancy of the volume:**

```
vxplex att vol01 vol01-01
```

9. **Remove the temporary volume:**

```
vxedit rm voltemp
```

For information on an alternative online backup method using the `vxassist` command, see Section 4.7, “Performing Online Backup.”

4.4.3 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, type:

```
vxplex att volume_name plex_name
```

For example, to associate a plex labeled `vol01-02` with a volume labeled `vol01` you would type:

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

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

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

For example, to create a mirrored, `fsgen`-type volume labeled `home` and associate two existing plexes labeled `home-1` and `home-2` you would type:

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

Note – You can also use the following command on an existing volume to add and associate a plex: `vxassist mirror` *volume_name*

4.4.4 Dissociating and Removing Plexes

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

- Providing free disk space
- Reducing the number of mirrors in a volume 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 – To save the data on a plex that is to be removed, you must identify the original configuration of that plex. Several parameters from that configuration, such as stripe unit size and subdisk ordering, are critical to the construction of a new plex that would contain the same data. Before such a plex is removed, you should record its configuration.

To dissociate and remove a plex from the associated volume, type:

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

To dissociate and remove a plex labeled `vol01-02`, type:

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

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

Note – You can first dissociate the plex and subdisks, then remove them with the commands:
`vxplex dis` *plex_name*
`vxedit -r rm` *plex_name*
Together, these commands accomplish the same thing as the `vxplex -o rm dis` command

4.4.5 Displaying Plex Information

To display detailed information about all plexes, type:

```
vxprint -lp
```

To display detailed information about a specific plex, type:

```
vxprint -l plex_name
```

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 plexes; the `–t` option prints a single line of information about the plex. To list free plexes, type:

```
vxprint -pt
```

4.4.6 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 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

Field	Description
`putil0`	Reserved for use by VxVM utilities and is retained on reboot.
`putil1`	Reserved for use by high-level utilities such as the Visual Administrator interface. This field is retained on reboot.
`putil2`	Reserved for use by the system administrator or site-specific applications. This field is retained on reboot.
`tutil0`	Reserved for use by VxVM utilities and is cleared on reboot.
`tutil1`	Reserved for use by high-level utilities such as the Visual Administrator interface. This field is cleared on reboot.
`tutil2`	Reserved for use by the system administrator or site-specific applications. This field is cleared on reboot.

To change plex attributes, type:

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

The command:

```
vxedit set comment="my plex" tutil2="u" user="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
```

4.4.7 Changing Plex 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 plex `OFFLINE` so that repair or maintenance can be performed on the physical disk containing that plex's subdisks, type:

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

If a disk drive suffered a head crash, put all plexes that have associated subdisks represented on the affected drive `OFFLINE`. For example, if plexes `vol01-02` and `vol02-02` had subdisks on a drive to be repaired, you would type:

```
vxmend off vol01-02 vol02-02
```

This command places `vol01-02` and `vol02-02` in the `OFFLINE` state, and they remain in that state until explicitly changed.

4.4.7.1 Detaching Plexes

To temporarily detach one plex in a mirrored volume, type:

```
vxplex det plex_name
```

For example, to temporarily detach a plex labeled `vol01-02` and place it in maintenance mode you would type:

```
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 contents of the plex will be recovered and it will be 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 by typing

```
vxvol start volume_name
```

If it is unstartable, set one of the plexes to CLEAN by typing:

```
vxmend fix clean plex_name
```

and then start the volume.

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

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

As with returning an OFFLINE plex to ACTIVE, this command recovers the contents of the plex(es), then sets the plex state to ACTIVE.

4.4.7.2 Attaching Plexes

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

If the volume is currently ENABLED, type:

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

For example, for a plex labeled `vol01-02` on a volume labeled `vol01` you would type:

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

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

If the volume is not in use (not ENABLED), type:

```
vxmend on plex_name
```

For example, for a plex labeled vol01-02 you would type:

```
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 you must manually change the state of a plex, refer to Section 4.5.10, "Volume Recovery." See the vxmake(1M) and vxmend(1M) manual pages for more information about these commands.

4.4.8 Moving Plexes

Moving a plex copies the data content from the original plex onto a new plex. 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, the copy of the data on the original plex will be incomplete. If this is what you want, then use the -o force option. 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 and will remain unused.

To move data from one plex to another type:

```
vxplex mv original_plex new_plex
```

4.4.9 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, type:

```
vxplex cp volume_name new_plex
```

After the copy operation is complete, *new_plex* will not be associated with the specified volume *volume_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 the same reason, *new_plex* also should not be sparse.

4.5 Volume Operations

A *volume* is a collection of up to 32 plexes that is shown as a block device in the `/dev/vx/dsk` directory and as a character device in the `/dev/vx/rdsk` directory. A volume can be used as a partition device.

Volume operations include:

- Creating a volume
- Initializing a volume
- Removing a volume
- Displaying volumes
- Changing volume attributes
- Resizing a volume
- Changing volume read policy
- Starting and stopping volumes
- Listing unstartable volumes
- Mirroring an existing volume
- Displaying plexes within a volume
- Recovering a disabled volume

The following sections describe how to perform common volume operations. In some cases, you can use either `vxassist` or one or more other commands to accomplish the same task.Both approaches are described. For detailed descriptions about the commands used to perform volume operations refer to the VxVM manual pages.

Note – This section provides information on general volume operations. For details on operations specific to RAID-5 volumes, see Section 4.6, “RAID-5 Volume Operations.”

4.5.1 Creating Volumes

You can use either `vxassist` or `vxmake` to create volumes.

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

4.5.1.1 `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, type:

```
vxassist make volume_name length
```

For example, to creates a volume named `voldef` with a length of 10 megabytes on any available disk(s), you would type:

```
vxassist make voldef 10m
```

You can specify additional parameters to `vxassist` to reflect the new volume’s attributes. Refer to the `vxassist`(1M) manual page for details.

The following example shows how to create 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
```

4.5.1.2 vxmake

To create a volume with an attached plex using `vxmake`, type:

```
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 an `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 as either `-Ufsgen` or `use_type=fsgen`. If a length is not specified or associated plexes are not identified, the length will be 0.

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 plexes, 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, see Section 4.2.4, “Creating VxVM Objects With vxmake” or refer to the `vxmake`(1M) manual page.

4.5.2 Initializing Volumes

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

```
vxvol init state volume_name [plex_name]
```

where the *state* variable determines what the initialization does and what condition the volume and plexes 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 plex (mirror), 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 *plex_name* argument. To set the state of this volume to `CLEAN`, type:

```
vxvol init clean volume_name
```

Note – The rest of this initialization section applies only to mirrored volumes (not RAID-5 volumes, which are discussed later).

Under more complicated circumstances, where a newly created volume `vol01` has multiple mirrors associated with it, one of the plexes must be chosen to which the other plexes are synchronized. For instance, if plex `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 plexes 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 plexes to `STALE`, so that they will be properly synchronized at the time the volume is made available. The administrator may avoid the initial synchronization of the

volume to save time, but only when certain that none of the plexes contain data that will be the final contents of the volume. Under such a situation, you can temporarily initialize the state of the volume so that the data can be loaded without having to perform a synchronization first by typing:

```
vxvol init enable volume_name
```

This enables the volume and all its plexes, but leaves the plex 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. You can initialize a volume (such as `home1`) for use and start it at the same time by typing:

```
vxvol init active home1
```

Caution – The data on each of the mirrors should be exactly the same under these circumstances. Otherwise, the system may corrupt the data on both mirrors and, depending on the use for the volume, could crash the system. If you are not sure that the data is identical, then use the `vxvol init clean` method.

Sometimes you must 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 typing:

```
vxvol init zero volume_name
```

4.5.3 Estimating Maximum Volume Size

The `vxassist` command can provide information on the largest possible size for a volume that can currently be created with a given set of attributes. `vxassist` can also provide similar information for how much an existing volume can be extended under the current circumstances.

To determine the largest possible size for the volume to be created, type:

```
vxassist maxsize attributes...
```

For example, to determine the maximum size for a new RAID-5 volume on available disks, type:

```
vxassist maxsize layout=raid5
```

This does not actually create the volume, but returns output such as the following (in sectors, by default):

```
Maximum volume size: 376832 (184Mb)
```

If, however, a volume with the specified attributes cannot be created, an error similar to the following might result:

```
vxvm:vxassist: ERROR: No volume can be created within the given constraints
```

To determine how much an existing volume can grow, type:

vxassist maxgrow *volume_name*

For example, the command:

```
vxassist maxgrow raidvol
```

does not actually resize the volume, but results in output similar to the following:

```
Volume raidvol can be extended by 366592 to 1677312 (819Mb)
```

Notice that this output indicates both the amount *by* which the volume can be increased and the total size *to* which the volume can grow (in sectors, by default).

4.5.4 Removing Volumes

Removing a volume is the same as removing a physical disk partition on a system without VxVM. 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 ensure that this data is retained.

To remove a volume, type:

```
vxedit rm volume_name
```

or

```
vxedit -rf rm volume_name
```

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

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

4.5.5 Displaying Volume Information

You can list information related to volumes under VxVM control, such as the name of the volume, its usage type, state, length, user and group IDs, and mode.

To list information on all volumes, type:

```
vxprint -vt
```

To limit the display of `vxprint` to a single object, specify the object name after the `vxprint` command.

For example, to display detailed information about a specific volume, type:

```
vxprint -l volume_name
```

If no volume is specified, detailed information is given for all volumes by typing:

```
vxprint -vl
```

To display the plexes for a volume labeled `vol01`, type:

```
vxprint -vt vol01
```

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

You can also display the plexes for `vol01` by typing:

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

4.5.6 Changing Volume Attributes

You can change volume attributes such as read policy, error policies, ownership, permissions, and the values in the comment and utility fields for existing volumes when the volume's use 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.

Use `vxedit` as follows:

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

Use `vxvol` as follows:

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

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

TABLE 4-2

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
`vxvol`	`len`	Numeric length of volume
	`logtype`	(`drl/undef`) specifier of Dirty Region Logging mode for volume
	`loglen`	Length of the Dirty Region Logging log
	`startopts`	Options to be executed to the `vxvol start` operation

Note – Do not use the `chgrp`, `chown`, and `chmod` commands to set volume device permissions and ownership. You must 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, type:

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

4.5.7 Resizing Volumes

To resize a volume, you change the volume attributes using either `vxassist` or `vxvol`.

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

You can use the `vxresize` command to resize a volume containing a file system. Although other commands can be used to resize volumes containing file systems, `vxresize` offers the advantage of automatically resizing the file system as well as the volume. For details on how to use `vxresize`, see the `vxresize`(1M) manual page. Note that only `vxfs` and `ufs` file systems can be resized with `vxresize`.

4.5.7.1 `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

Caution – If the volume contains a file system, do not shrink the volume below the size of the file system, as doing so could result in unrecoverable data loss. If you have a `vxfs` file system, you can shrink the file system before shrinking the volume.

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

To increase a volume to a specified length, type:

```
vxassist growto volume_name new_length
```

To increase a volume by a certain amount, type:

```
vxassist growby volume_name length_change
```

To reduce a volume to a specified length, type:

```
vxassist shrinkto volume_name new_length
```

To reduce a volume by a certain amount, type:

```
vxassist shrinkby volume_name length_change
```

4.5.7.2 vxvol

To change the length of a volume using `vxvol set`, type:

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

For example, to change the length to 100000 sectors, type:

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

Note – You cannot use the `vxvol set len` command to increase the size of a volume unless the needed space is available in the plexes of the volume. When you reduce a volume's size using the `vxvol set len` command, the freed space is not released into the free space pool.

4.5.7.3 Changing Volume Read Policy

VxVM offers the choice of the following read policies:

- `round` reads each plex in turn in "round-robin" fashion for each non-sequential I/O detected. Sequential access causes only one plex to be accessed, thus taking advantage of the drive or controller read-ahead caching policies.
- `prefer` reads preferentially from a plex that has been labeled as the preferred plex.
- `select` chooses a default policy based on plex associations to the volume. If the volume has an enabled striped plex, this defaults to preferring that plex; otherwise, it defaults to round-robin.

The read policy can be changed from `round` to `prefer` (or vice versa), or to a different preferred plex. The `vxvol rdpol` command sets the read policy for a volume.

To set the read policy to `round`, type:

```
vxvol rdpol round volume_name
```

For example, to set the read policy for volume `vol01` to a round-robin read, type:

```
vxvol rdpol round vol01
```

To set the read policy to `prefer`, type:

```
vxvol rdpol prefer volume_name preferred_plex_name
```

For example, to set the policy for `vol01` to read preferentially from the plex `vol01-02` type:

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

To set the read policy to `select`, type:

```
vxvol rdpol select volume_name
```

4.5.8 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, type:

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

To start all `DISABLED` volumes, type:

```
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, type:

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

For example, to stop a volume labeled `vol01` type:

```
vxvol stop vol01
```

To stop all `ENABLED` volumes, type:

```
vxvol stopall
```

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

```
vxvol maint volume_name
```

To assist in choosing the revival source plex, list the unstarted volume and displays its plexes.

To take plex `vol01-02` offline, type:

```
vxmend off vol01-02
```

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

```
vxmend on vol01-02
```

4.5.8.1 Listing Unstartable Volumes

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

```
vxinfo [volume_name]
```

4.5.9 Mirroring Existing Volumes

You can add a mirror (plex) to an existing volume using the `vxassist` command:

```
vxassist mirror volume_name
```

For example, to create a mirror of the volume `voltes, type`:

```
vxassist mirror voltest
```

Another way to mirror an existing volume is by first creating a plex and then associating it with a volume, by typing:

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

4.5.10 Volume Recovery

If a system crash or an I/O error corrupts one or more plexes of a volume and no plex is `CLEAN` or `ACTIVE`, mark one of the plexes `CLEAN` and instruct the system to use that plex as the source for reviving the others.

To place a plex in a `CLEAN` state, type:

```
vxmend fix clean plex_name
```

For example, to place the plex labeled `vol01-02` in the `CLEAN` state, type:

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

For information about how to use `vxmend`, refer to the `vxmend`(1M) manual page. For general information on recovery, see to Appendix B, “Recovery.”

4.6 RAID-5 Volume Operations

This section discusses the construction and operation of RAID-5 volumes. For more information on RAID-5 volumes, refer to Chapter 1.

RAID-5 and mirrored volumes both maintain redundancy of the data within a volume; mirrored volumes do so by keeping multiple, complete copies of the data in different plexes, whereas RAID-5 volumes keep one copy of the data and calculated parity in the same plex.

Both mirrored and RAID-5 volumes have the ability to do logging to minimize recovery time. Mirrored volumes use dirty region logs to keep a map of which regions of the volume are currently being written, while RAID-5 volumes use RAID-5 logs to keep a copy of the data and parity currently being written.

4.6.1 RAID-5 Volume Layout

Like a mirrored volume, a RAID-5 volume is made up of one or more plexes, each of which consists of one or more subdisks. Unlike mirrored volumes, not all plexes in a RAID-5 volume serve to keep a mirror copy of the volume data. A RAID-5 volume can have two types of plexes:

- The *RAID-5 plex* is used to keep both data and parity for the volume.
- *Log plexes* keep logs of data written to the volume for faster recovery and increased resilience in the face of failures.

4.6.1.1 RAID-5 Plexes

RAID-5 volumes keep both the data and parity information in a single RAID-5 plex. A RAID-5 plex is made up of subdisks arranged in columns, similar to the striping model. FIGURE 4-1 shows a RAID-5 plex and the output of `vxprint` that would be associated with it:

```
PL   NAME      VOLUME   KSTATE  STATE    LENGTH   LAYOUT    NCOL/WID  MODE
SD   NAME      PLEX     DISK    DISKOFFS LENGTH   [COL/]OFF DEVICE    MODE

pl   rvol-01   rvol     ENABLED ACTIVE   20480    RAID      3/16      RW
sd   disk00-00 rvol-01  disk00  0        10240    0/0       c1t4d1    ENA
sd   disk01-00 rvol-01  disk01  0        10240    1/0       c1t2d1    ENA
sd   disk02-00 rvol-01  disk02  0        10240    2/0       c1t3d1    ENA
```

FIGURE 4-1 `vxprint` Output for a RAID-5 Plex

The plex line shows that the plex layout is `RAID` and that it has three columns and a stripe unit size of 16 sectors. Each subdisk line shows the column in the plex and offset in the column in which it is located.

4.6.1.2 RAID-5 Logs

Each RAID-5 volume can have at most one RAID-5 plex where the data and parity will be stored. Any other plexes associated with the volume are used to log information about data and parity being written to the volume. These plexes are referred to as *RAID-5 log plexes* or *RAID-5 logs***.**

RAID-5 logs can be concatenated or striped plexes, and each RAID-5 log associated with a RAID-5 volume will have a complete copy of the logging information for the volume. It is suggested that you have a minimum of two RAID-5 log plexes for each RAID-5 volume. These log plexes should be located on different disks. Having two RAID-5 log plexes for each RAID-5 volume protects against the loss of logging information due to the failure of a single disk.

To support enough concurrent access to the RAID-5 array, the log should be several times the stripe size of the RAID-5 plex.

RAID-5 log plexes can be differentiated from the RAID-5 plex of a RAID-5 volume by examining `vxprint` output; the `STATE` field for a log plex is marked as `LOG`. FIGURE 4-2 shows the `vxprint` output for a RAID-5 volume.

```
V   NAME       USETYPE  KSTATE   STATE    LENGTH   READPOL   PREFPLEX
PL  NAME       VOLUME   KSTATE   STATE    LENGTH   LAYOUT    NCOL/WID   MODE
SD  NAME       PLEX     DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE     MODE

v   r5vol      raid5    ENABLED  ACTIVE   20480    RAID      -
pl  r5vol-01   r5vol    ENABLED  ACTIVE   20480    RAID      3/16       RW
sd  disk00-00  r5vol-01 disk00   0        10240    0/0       c1t4d1     ENA
sd  disk01-00  r5vol-01 disk01   0        10240    1/0       c1t2d1     ENA
sd  disk02-00  r5vol-01 disk02   0        10240    2/0       c1t3d1     ENA
pl  r5vol-l1   r5vol    ENABLED  LOG      1024     CONCAT    -          RW
sd  disk03-01  r5vol-l1 disk00   0        1024     0         c1t3d0     ENA
pl  r5vol-l2   r5vol    ENABLED  LOG      1024     CONCAT    -          RW
sd  disk04-01  r5vol-l2 disk02   0        1024     0         c1t1d1     ENA
```

FIGURE 4-2 `vxprint` Output for a RAID-5 Volume

The RAID-5 volume (`r5vol`) can be identified as a RAID-5 volume by its read policy being `RAID`. It has one RAID-5 plex (`r5vol-01`), similar to the one described earlier. It has two RAID-5 logs in the plexes `r5vol-l1` and `r5vol-l2`; these are identified by the state field being `LOG` and the fact that they are associated with a RAID-5 volume and have a layout that is not `RAID`.

4.6.2 Creating RAID-5 Volumes

You can create RAID-5 volumes using either `vxassist` (the recommended approach) or `vxmake`. Both approaches are discussed in this section.

A RAID-5 volume contains a RAID-5 plex that consists of two or more subdisks located on two or more physical disks. Only one RAID-5 plex can exist per volume.

4.6.2.1 `vxassist`

To create a RAID-5 volume using `vxassist`, type:

```
vxassist make volume_name length layout=raid5
```

For example, to create a 10M RAID-5 volume named `volraid`, type:

```
vxassist make volraid 10m layout=raid5
```

This creates a RAID-5 volume with the default stripe unit size on the default number of disks.

4.6.2.2 vxmake

Creating RAID-5 volumes through the `vxmake` command is similar to creating other volumes (see Section 4.5.1, "Creating Volumes" and the `vxmake`(1M) manual page for details on creating other volumes). Creating subdisks for use in a RAID-5 volume is the same as creating any subdisk.

The procedure for creating a RAID-5 plex for a RAID-5 volume is similar to that for creating striped plexes, except that the `layout` attribute should be set to `raid5`. Subdisks can be implicitly associated in the same way as with striped plexes. Thus, you can create a four-column RAID-5 plex with a stripe unit size of 32 sectors from four existing subdisks by typing:

```
vxmake plex raidplex layout=raid5 stwidth=32 \
sd=disk00-01,disk01-00,disk02-00,disk03-00
```

Note that because four subdisks were specified without any specification of columns, `vxmake` assumes a four-column RAID-5 plex and places one subdisk in each column. This is the same thing that happens when creating striped plexes. If the subdisks are to be created later, you could use the following command to create the plex:

```
vxmake plex raidplex layout=raid5 ncolumn=4 stwidth=32
```

Note – If no subdisks are specified, you must specify the `ncolumn` attribute. You can fill in the plex with subdisks later by using `vxsd assoc` (see Section 4.6.6.2, "Manipulating RAID-5 Subdisks").

To create a three-column RAID-5 plex using six subdisks, you could type:

```
vxmake plex raidplex layout=raid5 stwidth=32 \
sd=disk00-00:0,disk01-00:1,disk02-00:2,disk03-00:0, \
disk04-00:1,disk05-00:2
```

This would stack subdisks `disk00-00` and `disk03-00` consecutively in column 0, subdisks `disk01-00` and `disk04-00` consecutively in column 1, and subdisks `disk02-00` and `disk05-00` in column 2. You can also specify offsets to create sparse RAID-5 plexes, as for striped plexes.

Since log plexes are simply plexes without a RAID-5 layout, they can be created normally.

To create a RAID-5 volume with `vxmake`, simply specify the usage type to be `RAID-5`:

```
vxmake -Uraid5 vol raidvol
```

RAID-5 plexes and RAID-5 log plexes can be associated implicitly:

```
vxmake -Uraid5 vol raidvol plex=raidplex,raidlog1, raidlog2
```

4.6.3 Initializing RAID-5 Volumes

You need to initialize a RAID-5 volume if it was created by `vxmake` and has not yet been initialized or if it has somehow been set to an uninitialized state.

You can initialize a RAID-5 volume through `vxvol` in either of the following ways:

```
vxvol init zero volume_name
```

or

```
vxvol start volume_name
```

`vxvol init zero` writes zeroes to any RAID-5 log plexes and to the entire length of the volume; it then leaves the volume in the `ACTIVE` state.

`vxvol start` recovers parity by XORing together corresponding data stripe units in all other columns. Although it is slower than a `vxvol init zero` operation, `vxvol start` makes the RAID-5 volume immediately available.

4.6.4 Failures and RAID-5 Volumes

Failures, when taken in the context of volume management, come in two varieties: *system failures* and *disk failures*. A system failure means that the system has abruptly ceased to operate, such as due to an operating system panic or power failure. Disk failures imply that the data on some number of disks has become unavailable due to some sort of failure in the system (such as a head crash, electronics failure on disk, or disk controller failure).

4.6.4.1 System Failures

RAID-5 volumes are designed to remain available in the face of disk failures with a minimum of disk space overhead. However, many implementations of RAID-5 can become vulnerable to data loss after a system failure. This occurs because a system failure causes the data and parity in the RAID-5 volume to become unsynchronized because the disposition of writes that were outstanding at the time of the failure cannot be determined. If this occurs while a RAID-5 volume is being accessed, the volume is described as having *stale parity*. When this occurs, the parity must be reconstructed by reading all the non-parity columns within each stripe, recalculating the parity, and writing out the parity stripe unit in the stripe. This must be done for every stripe in the volume, so it can take a long time to complete.

Caution – While this resynchronization is going on, any failure of a disk within the array will cause the data in the volume to be lost. This only applies to RAID-5 volumes *without* log plexes.

Having the array vulnerable in this way is undesirable. Besides the vulnerability to failure, the resynchronization process can tax the system resources and slow down system operation.

RAID-5 logs reduce the possible damage that can be caused by system failures. Because they maintain a copy of the data being written at the time of the failure, the process of resynchronization consists of simply reading that data and parity from the logs and writing it to the appropriate areas of the RAID-5 volume. This greatly reduces the amount of time needed for a resynchronization of data and parity. It also means that the volume never becomes truly stale because the data and parity for all stripes in the volume is known at all times, so the failure of a single disk cannot result in the loss of the data within the volume.

4.6.4.2 Disk Failures

Disk failures can cause the data on a disk to become unavailable. In terms of a RAID-5 volume, this would mean that a subdisk becomes unavailable. This can occur due to an uncorrectable I/O error during a write to the disk (which causes the subdisk to be detached from the array) or due to a disk being unavailable when the system is booted (such as from a cabling problem or having a drive powered down). When this occurs, the subdisk cannot be used to hold data and is considered *stale* and *detached*. If the underlying disk becomes available or is replaced, the subdisk will still be considered stale and will not be used.

If an attempt is made to read data contained on a stale subdisk, the data is reconstructed from data from all other stripe units in the stripe; this operation is called a *reconstructing-read*. This is a significantly more expensive operation than simply reading the data, resulting in degraded read performance; thus, when a RAID-5 volume has stale subdisks, it is considered to be in *degraded mode*.

A RAID-5 volume in degraded mode can be recognized as such from the output of `vxprint`, as shown in FIGURE 4-3.

```
V    NAME       USETYPE   KSTATE   STATE    LENGTH  READPOL   PREFPLEX
PL   NAME       VOLUME    KSTATE   STATE    LENGTH  LAYOUT    NCOL/WID MODE
SD   NAME       PLEX      DISK     DISKOFFS LENGTH  [COL/]OFF DEVICE   MODE
v    r5vol      RAID-5    ENABLED  DEGRADED 20480   RAID      -
pl   r5vol-01   r5vol     ENABLED  ACTIVE   20480   RAID      3/16     RW
sd   disk00-00  r5vol-01  disk00   0        10240   0/0       c1t4d1
sd   disk01-00  r5vol-01  disk01   0        10240   1/0       c1t2d1   dS
sd   disk02-00  r5vol-01  disk02   0        10240   2/0       c1t3d1   -
pl   r5vol-l1   r5vol     ENABLED  LOG      1024    CONCAT    -        RW
sd   disk03-01  r5vol-l1  disk00   10240    1024    0         c1t3d0   -
pl   r5vol-l2   r5vol     ENABLED  LOG      1024    CONCAT    -        RW
sd   disk04-01  r5vol-l2  disk02   10240    1024    0         c1t1d1   -
```

FIGURE 4-3 `vxprint` Output for a Degraded RAID-5 Volume

The volume `r5vol` is in degraded mode, as shown by the `STATE`, which is listed as `DEGRADED`. The failed subdisk is `disk01-00`, as shown by the flags in the last column — the `d` indicates that the subdisk is detached and the `S` indicates that the subdisk contents are stale.

It is also possible that a disk containing a RAID-5 log could experience a failure. This has no direct effect on the operation of the volume; however, the loss of all RAID-5 logs on a volume makes the volume vulnerable to a complete failure, as described

earlier. In the output of `vxprint -ht`, failure within a RAID-5 log plex is indicated by the plex state being `BADLOG`, as shown in FIGURE 4-4, where the RAID-5 log plex `r5vol-l1` has failed.

```
V    NAME        USETYPE    KSTATE   STATE    LENGTH  READPOL   PREFPLEX
PL   NAME        VOLUME     KSTATE   STATE    LENGTH  LAYOUT    NCOL/WID MODE
SD   NAME        PLEX       DISK     DISKOFFS LENGTH  [COL/]OFF DEVICE   MODE
v    r5vol       RAID-5     ENABLED  ACTIVE   20480   RAID      -
pl   r5vol-01    r5vol      ENABLED  ACTIVE   20480   RAID      3/16     RW
sd   disk00-00   r5vol-01   disk00   0        10240   0/0       c1t4d1   ENA
sd   disk01-00   r5vol-01   disk01   0        10240   1/0       c1t2d1   dS
sd   disk02-00   r5vol-01   disk02   0        10240   2/0       c1t3d1   ENA
pl   r5vol-l1    r5vol      DISABLED BADLOG   1024    CONCAT    -        RW
sd   disk03-01   r5vol-l1   disk00   10240    1024    0         c1t3d0   ENA
pl   r5vol-l2    r5vol      ENABLED  LOG      1024    CONCAT    -        RW
sd   disk04-01   r5vol-l2   disk02   10240    1024    0         c1t1d1   ENA
```

FIGURE 4-4 `vxprint` Output for a RAID-5 Volume with Failed Log Plex

4.6.5 RAID-5 Recovery

The types of recovery typically needed for RAID-5 volumes are:

- Parity resynchronization
- Stale subdisk recovery
- Log plex recovery

These types of recoveries are discussed in the sections that follow. Parity resynchronization and stale subdisk recovery are typically performed when the RAID-5 volume is started, shortly after the system boots, or by calling the `vxrecover` command. For more information on starting RAID-5 volumes, see Section 4.6.6.3, “Starting RAID-5 Volumes.”

If hot-relocation is enabled at the time of a disk failure, system administrator intervention is not required (unless there is no suitable disk space available for relocation). Hot-relocation is triggered by the failure and the system administrator is notified of the failure by electronic mail. Hot-relocation automatically attempts to relocate the subdisks of a failing RAID-5 plex. After any relocation takes place, the hot-relocation daemon (`vxrelocd`) also initiates a parity resynchronization. In the case of a failing RAID-5 log plex, relocation occurs only if the log plex is mirrored; `vxrelocd` then initiates a mirror resynchronization to re-create the RAID-5 log plex. If hot-relocation is disabled at the time of a failure, the system administrator may need to initiate a resynchronization or recovery.

4.6.5.1 Parity Resynchronization

In most circumstances, a RAID-5 array does not have stale parity. Stale parity should only occur after all RAID-5 log plexes for the RAID-5 volume have failed, and then only if there is a system failure. Furthermore, even if a RAID-5 volume has stale parity, it is usually taken care of as part of the volume start process. However, if a volume without valid RAID-5 logs is started and the process is killed before the volume is resynchronized, the result will be an active volume with stale parity. This can be confirmed by checking the volume state displayed in the output of the `vxprint -ht` command, as shown in FIGURE 4-5.

```
V   NAME       USETYPE   KSTATE   STATE    LENGTH  READPOL   PREFPLEX
PL  NAME       VOLUME    KSTATE   STATE    LENGTH  LAYOUT    NCOL/WID MODE
SD  NAME       PLEX      DISK     DISKOFFS LENGTH  [COL/]OFF DEVICE   MODE
v   r5vol      RAID-5    ENABLED  NEEDSYNC 20480   RAID      -
pl  r5vol-01   r5vol     ENABLED  ACTIVE   20480   RAID      3/16     RW
sd  disk00-00  r5vol-01  disk00   0        10240   0/0       c1t4d1   ENA
sd  disk01-00  r5vol-01  disk01   0        10240   1/0       c1t2d1   ENA
sd  disk02-00  r5vol-01  disk02   0        10240   2/0       c1t3d1   ENA
```

FIGURE 4-5 `vxprint` Output for a Stale RAID-5 Volume

This output lists the volume state as `NEEDSYNC`, indicating that the parity needs to be resynchronized. The state could also have been `SYNC`, indicating that a synchronization was attempted at start time and that a synchronization process should be doing the synchronization. If no such process exists or if the volume is in the `NEEDSYNC` state, you can manually start a synchronization using the `resync` keyword for the `vxvol` command. For example, to resynchronize the RAID-5 volume in FIGURE 4-5, you would type:

```
vxvol resync r5vol
```

Parity is regenerated by issuing `VOL_R5_RESYNC` ioctls to the RAID-5 volume. The resynchronization process starts at the beginning of the RAID-5 volume and resynchronizes a region equal to the number of sectors specified by the `-o` *iosize* option or, if `-o` *iosize* is not specified, the default maximum I/O size. The `resync` operation then moves onto the next region until the entire length of the RAID-5 volume has been resynchronized.

For larger volumes, parity regeneration can take a significant amount of time and it is possible that the system could be shut down or crash before the operation is completed. Unless the progress of parity regeneration is somehow kept across reboots, the process would have to start all over again. To avoid this situation, parity regeneration is *checkpointed*, meaning that the offset up to which the parity has been regenerated is saved in the configuration database. The `-o checkpt=`*size* option

controls how often the checkpoint is saved; if not specified, it defaults to the default checkpoint size. Since saving the checkpoint offset requires a transaction, making the checkpoint size too small can significantly extend the time required to regenerate parity. After a system reboot, a RAID-5 volume that has a checkpoint offset smaller than the volume length will start a parity resynchronization at the checkpoint offset.

4.6.5.2 Subdisk Recovery

Like parity resynchronization, stale subdisk recovery is usually done at volume start time. However, it is possible that the process doing the recovery may get killed, or that the volume was started with an option to prevent subdisk recovery. It's also possible that the disk on which the subdisk resides was replaced without any recovery operations being performed. In any case, a subdisk recovery can be achieved by using the `recover` keyword of the `vxvol` command. For example, to recover the stale subdisk in the RAID-5 volume shown in FIGURE 4-3, you would type:

```
vxvol recover r5vol disk01-00
```

If a RAID-5 volume has multiple stale subdisks to be caught up all at once, calling `vxvol recover` with only the name of the volume will achieve this:

```
vxvol recover r5vol
```

4.6.5.3 Log Plex Recovery

RAID-5 log plexes may become detached due to disk failures, as shown in FIGURE 4-4. These RAID-5 logs can be reattached by using the `att` keyword for the `vxplex` command. To reattach the failed RAID-5 log plex shown in FIGURE 4-4, type:

```
vxplex att r5vol r5vol-l1
```

4.6.6 Miscellaneous RAID-5 Operations

Many operations are available for manipulating RAID-5 volumes and associated objects. These operations are usually performed by other commands such as `vxassist` and `vxrecover` as part of larger operations, such as evacuating disks, etc. These command line operations should not be necessary for casual usage of the Volume Manager.

4.6.6.1 Manipulating RAID-5 Logs

RAID-5 logs are represented as plexes of RAID-5 volumes and are manipulated using the `vxplex` command. You can add a RAID-5 log using `vxplex att`:

```
vxplex att r5vol r5log
```

The attach operation can only proceed if the new log's size is large enough to hold an entire stripe's worth of data. If the RAID-5 volume already has logs, the new log must be at least as large as the volume's log length; otherwise, the volume log length is set to the `contig len` of the new log plex.

If the RAID-5 volume is not enabled, the new log is marked as `BADLOG` and will be enabled when the volume is started, though its contents will be ignored. If the RAID-5 volume is enabled and has other enabled RAID-5 logs, the new log will have its contents synchronized with the other logs via `ATOMIC_COPY` ioctls. If the RAID-5 volume currently has no enabled logs, the new log is zeroed before it is enabled.

You can remove log plexes from a volume using the `vxplex dis` command:

```
vxplex dis r5log3
```

If removing the log leaves the volume with less than two valid logs, a warning is printed and the operation is stopped. You must force the operation by using the `-o force` option.

4.6.6.2 Manipulating RAID-5 Subdisks

Like other subdisks, subdisks of the RAID-5 plex of a RAID-5 volume are manipulated using the `vxsd` command. Association is done using the `assoc` keyword in the same manner as for striped plexes. For example, to add subdisks at the end of each column of the RAID-5 volume in FIGURE 4-2, you would type:

```
vxsd assoc r5vol-01 disk10-01:0 disk11-01:1 disk12-01:2
```

If a subdisk is filling a "hole" in the plex (that is, some portion of the volume's logical address space is mapped by the subdisk), the subdisk is considered stale. If the RAID-5 volume is enabled, the association operation regenerates the data that belongs on the subdisk using `VOL_R5_RECOVER` ioctls; otherwise, it is simply marked as stale and is recovered when the volume is started.

Subdisks can be removed from the RAID-5 plex by using `vxsd dis`:

```
vxsd dis disk10-01
```

Caution – If the subdisk maps a portion of the RAID-5 volume's address space, this places the volume in `DEGRADED` mode. If this is the case, the `dis` operation prints a warning and must be forced using the `-o force` option. Additionally, if removing the subdisk makes the RAID-5 volume unusable (because another subdisk in the same stripe is unusable or missing) and the volume is not `DISABLED` and empty, this operation is not allowed.

Subdisks can be moved to change the disks that a RAID-5 volume occupies by using `vxsd mv`. For example, if you need to evacuate `disk03` and `disk22` has enough room by using two portions of its space, you could type:

```
vxsd mv disk03-01 disk22-01 disk22-02
```

While this command is similar to that for striped plexes, the actual mechanics of the operation are not. In order to do RAID-5 subdisk moves, the old subdisk is removed from the RAID-5 plex and replaced by the new subdisks, which are marked as stale and then recovered using `VOL_R5_RECOVER` operations either by `vxsd` or (if the volume is not active) when the volume is started. *This means that the RAID-5 volume is degraded for the duration of the operation*. Another failure in the stripes involved in the move will make the volume unusable. The RAID-5 volume could also become invalid if the parity of the volume were to become stale.

To avoid these situations, the `vxsd` utility will not allow a subdisk move if:

- A stale subdisk occupies any of the same stripes as the subdisk being moved
- The RAID-5 volume is stopped but was not shut down cleanly (parity is considered stale)
- The RAID-5 volume is active and has no valid log areas

Only the third case can be overridden using the `-o force` option.

Subdisks of RAID-5 volumes can also be split and joined by using `vxsd split` and `vxsd join`. These operations work the same as for mirrored volumes.

4.6.6.3 Starting RAID-5 Volumes

When a RAID-5 volume is started, it can be in one of many states. After a normal system shutdown, the volume should be clean and require no recovery. However, if the volume was not closed (or was not unmounted before a crash), it may require some type of recovery when it is started before it can be made available. This section outlines actions that should be taken under certain circumstances.

Under normal circumstances, volumes are started automatically after a reboot and any recovery takes place automatically or is done via the `vxrecover` command.

Unstartable RAID-5 Volumes

A RAID-5 volume is unusable if some part of the RAID-5 plex does not map the volume length. In other words, the RAID-5 plex cannot be sparse in relation to the RAID-5 volume length. The RAID-5 plex does not map a region where two subdisks have failed within a stripe, either because they are stale or because they are built on a failed disk. When this occurs, the `vxvol start` command returns the following error message:

```
vxvm:vxvol: ERROR: Volume r5vol is not startable; RAID-5 plex
does not map entire volume length.
```

At this point, the contents of the RAID-5 volume are unusable.

Another way that a RAID-5 volume can become unstartable is if the parity is stale and a subdisk becomes detached or stale. This occurs because within the stripes that contain the bad subdisk, the parity stripe unit is invalid (because the parity is stale) *and* the stripe unit on the bad subdisk is also invalid. This situation is shown in FIGURE 4-6, which illustrates a RAID-5 volume that has become invalid due to stale parity and a failed subdisk.

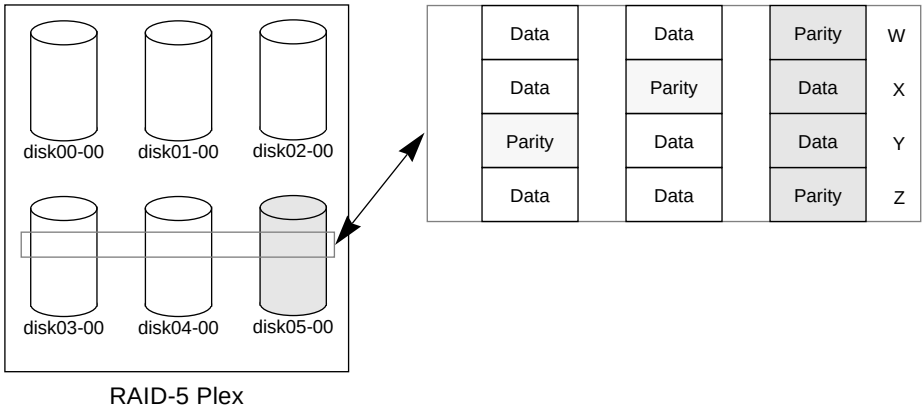


FIGURE 4-6 Invalid RAID-5 Volume

This example shows four stripes in the RAID-5 array. All parity is stale and subdisk `disk05-00` has failed. This makes stripes X and Y unusable because two failures have occurred within those stripes.

This qualifies as two failures within a stripe and prevents the use of the volume. In this case, the output from `vxvol start` would be:

```
vxvm:vxvol: ERROR: Volume r5vol is not startable; some subdisks
are unusable and the parity is stale.
```

You can avoid this situation by *always* using RAID-5 log plexes in RAID-5 volumes, preferably two or more. RAID-5 log plexes prevent the parity within the volume from becoming stale, thus preventing this situation (see Section 4.6.4.1, “System Failures,” for details).

Forcibly Starting RAID-5 Volumes

You may want to start a volume despite the fact that subdisks are marked as stale. For example, if a stopped volume has stale parity and no RAID-5 logs and a disk becomes detached and then reattached, the subdisk is considered stale even though the data is not out of date (because the volume was in use when the subdisk was unavailable) and the RAID-5 volume is considered invalid. The best way to prevent this case is to always have multiple valid RAID-5 logs associated with the array. However, this may not always be possible.

To start a RAID-5 volume despite the presence of stale subdisks, use the `-f` option with the `vxvol start` command. This causes all stale subdisks to be marked as non-stale before the `start` operation begins its evaluation of the validity of the RAID-5 volume and what is needed to start it. Alternately, individual subdisks can be marked as non-stale by using the `vxmend fix unstale` *subdisk* command.

4.6.6.4 Recovery When Starting RAID-5 Volumes

It may take several operations to fully restore the contents of a RAID-5 volume and make it usable. Whenever a volume is started, any RAID-5 log plexes are zeroed before the volume is started; this prevents random data from being interpreted as a log entry and corrupting the volume contents. Beyond RAID-5 log zeroing, some subdisks may need to be recovered, or the parity may need to be resynchronized (if RAID-5 logs have failed).

When a RAID-5 volume is started the following occurs:

1. If the RAID-5 volume was not cleanly shut down, it is checked for valid RAID-5 log plexes.
   - If valid log plexes exist, they are replayed. This is done by placing the volume in the `DETACHED` kernel state and setting the volume state to `REPLAY`, and enabling the RAID-5 log plexes. If the logs can be successfully read and the replay is successful, move on to Step 2.
   - If no valid logs exist, the parity must be resynchronized. Resynchronization is performed by placing the volume in the `DETACHED` kernel state and setting the volume state to `SYNC`. Any log plexes are left `DISABLED`.

     The volume is not made available while the parity is resynchronized because any subdisk failures during this period would make the volume unusable. You can override this by using the `-o unsafe` start option with `vxvol`.

Caution – The `-o unsafe` start option is considered dangerous, as it can make the contents of the volume unusable. It is therefore not recommended.

   If any stale subdisks exist, the RAID-5 volume is unusable.

2. Any existing logging plexes are zeroed and enabled. If all logs fail during this process, the start process is aborted.
3. If no stale subdisks exist or those that exist are recoverable, the volume is put in the `ENABLED` kernel state and the volume state is set to `ACTIVE`. The volume is now started.

4. If some subdisks are stale and need recovery, and if valid logs exist, the volume is enabled by placing it in the ENABLED kernel state and the volume is available for use during the subdisk recovery. Otherwise, the volume kernel state is set to DETACHED and it is not available during subdisk recovery.

This is done because if the system were to crash or the volume was ungracefully stopped while it was active, the parity would become stale, making the volume unusable. If this is undesirable, you can start the volume with the `-o unsafe` start option.

Caution – The `-o unsafe` start option can make the contents of the volume unusable.

5. The volume state is set to RECOVER and stale subdisks are restored. As the data on each subdisk becomes valid, the subdisk is marked as no longer stale.

If any subdisk recovery fails and there are no valid logs, the volume start is aborted because the subdisk would remain stale and a system crash would make the RAID-5 volume unusable. This can also be overridden by using the `-o unsafe` start option.

Caution – The `-o unsafe` start option can make the contents of the volume unusable.

If the volume has valid logs, subdisk recovery failures are noted but do not stop the start procedure.

6. When all subdisks have been recovered, the volume is placed in the ENABLED kernel state and marked as ACTIVE. It is now started.

4.6.6.5 Changing RAID-5 Volume Attributes

You can change several attributes of RAID-5 volumes. For RAID-5 volumes, the volume length and RAID-5 log length can be manipulated using the `vxvol set` command. To change the length of a RAID-5 volume, type:

```
vxvol set len=10240 r5vol
```

The length of a volume cannot exceed the mapped region (called the *contiguous length*, or *contiglen*) of the RAID-5 plex. In other words, the length cannot be extended so as to make the volume unusable. If the RAID-5 volume is active and the length is being shortened, you must force the operation using the `-o force` usage type option; this prevents yanking space away from applications using the volume.

You can also change the length of the RAID-5 logs using `vxvol set`

```
vxvol set loglen=2M r5vol
```

Remember that RAID-5 log plexes are only valid if they map the entire length of the RAID-5 volume's log length. If increasing the log length would make any of the RAID-5 logs invalid, the operation will not be allowed. Also, if the volume is not active and is dirty (that is, was not shut down cleanly), the log length cannot be changed. This avoids the loss of any of the log contents (if the log length is decreased) or the introduction of random data into the logs (if the log length is being increased).

4.7 Performing Online Backup

You can perform snapshot backups of volume devices using VxVM through `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 plex can be larger than necessary, but if a plex that is too small 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 plex which gets attached to and synchronized with the volume. When synchronized with the volume, the backup plex 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 the `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.

To complete the online backup procedure, run 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.

Use the following steps to perform a complete `vxassist` backup:

Note – This procedure does not apply to RAID-5.

---

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 ask 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
```

APPENDIX **A**

Volume Manager Error Messages

This appendix provides information on error messages associated with the Volume Manager configuration daemon (`vxconfigd`) and the kernel. It covers most informational, failure, and error messages displayed (on the console) by `vxconfigd` and the kernel driver. These include some errors that are infrequently encountered and difficult to troubleshoot.

The message descriptions are numbered for ease of reference. Clarifications are included to elaborate on the situation or problem that may have generated a particular message. Wherever possible, a recovery procedure (action) is provided to locate and correct the problem.

A.1 Logging Error Messages

The Volume Manager provides the option of logging console output to a file. This logging is useful in that any messages output just before a system crash will be available in the log file (presuming that the crash does not result in file system corruption). `vxconfigd` controls whether such logging is turned on or off. If enabled, the default log file is `/var/vxvm/vxconfigd.log`.

`vxconfigd` also supports the use of `syslog()` to log all of its regular console messages. When this is enabled, all console output is directed through the `syslog()` interface.

`syslog()` and log file logging can be used together to provide reliable logging (into a private log file), along with distributed logging through `syslogd`.

For Solaris, both `syslog()` and log file logging are disabled by default.

To enable logging of console output to a file, you can either invoke `vxconfigd` as follows or edit VxVM startup scripts (described later):

```
vxconfigd -x log
```

To enable `syslog()` logging of console output, you can either invoke `vxconfigd` as follows or edit VxVM startup scripts (described later):

```
vxconfigd -x syslog
```

To enable log file and/or `syslog()` logging, you can also edit the following portion of the `/etc/init.d/vxvm-sysboot` startup script:

```
# comment-out or uncomment any of the following lines to enable or
# disable the corresponding feature in vxconfigd.

#opts="$opts -x syslog"                # use syslog for console messages
#opts="$opts -x log"                   # messages to /var/vxvm/vxconfigd.log
#opts="$opts -x logfile=/foo/bar"      # specify an alternate log file
#opts="$opts -x timestamp"             # timestamp console messages

# to turn on debugging console output, uncomment the following line.
# The debug level can be set higher for more output.  The higest debug
# level is 9.

#debug=1                               # enable debugging console ouptut
```

Uncomment the line(s) corresponding to the feature(s) that you want enabled at startup. For example, to set up `vxconfigd` to automatically use file logging, uncomment the `opts="$opts -x log"` string.

For more information on logging options available through `vxconfigd`, refer to the `vxconfigd(1M)` manual page.

A.2 Volume Manager Configuration Daemon Error Messages

The Volume Manager is fault-tolerant and resolves most problems without system administrator intervention. If the Volume Manager configuration daemon (`vxconfigd`) recognizes what actions are necessary, it will queue up the transactions that are required. VxVM provides 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 error messages associated with the Volume Manager configuration daemon.

A.2.1 `vxconfigd` Usage Messages

The following are usage messages associated with `vxconfigd`.

A.2.1.1 `Usage: vxconfigd - long`

```
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 resetreset kernel state; requires 'reset' option argument
-m mode set vold's operating mode
    modes: disable, enable, bootload, bootstart
-x debugset debugging level to <debug>, 0 turns off debugging
-R fileset filename for client request rendezvous
-D fileset filename for client diag request rendezvous
```

- **Clarification**

This is the full usage message for `vxconfigd`, which is displayed when you type the `vxconfigd help` command.

A.2.1.2 Usage: vxconfigd - short

```
Usage: vxconfigd [-dkf] [-r reset] [-m mode] [-x level]
For detailed help use: vxconfigd help
```

- **Clarification**

  This is the standard `vxconfigd` usage error message. This message is displayed if some option was supplied incorrectly.

- **Action**

  If you need help in using `vxconfigd`, try using the command `vxconfigd help`.

  For more detailed information, see the `vxconfigd(1M)` manual page.

A.2.1.3 -r must be followed by 'reset'

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

- **Clarification**

  This is a usage error. The `-r` option requires an option argument consisting of the string `reset`.

- **Action**

  Either don't use the `-r` option, or supply the `reset` option argument.

A.2.1.4 -x argument: invalid debug string

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

- **Clarification**

  An unrecognized string was specified as an argument to the `-x` option.

- **Action**

  See `vxconfigd(1M)` for a list of valid arguments to `-x`.

A.2.1.5 -x devprefix=device_prefix: prefix too long

```
-x devprefix=device_prefix: prefix too long
```

- **Clarification**

  The `-x devprefix=`*device_prefix* option was used to define a prefix path for the `/dev/dsk` and `/dev/rdsk` directories, and that prefix was too long.

- **Action**

  Use a shorter prefix.

A.2.2 vxconfigd Error Messages

The following are general error messages associated with `vxconfigd`.

A.2.2.1 signal_name [core dumped]

```
vxvm:vxconfigd: ERROR: signal_name [ - core dumped ]
```

- **Clarification**

  The `vxconfigd` daemon encountered an unexpected signal while starting up. The specific signal is indicated by *signal_name*. If the signal caused the `vxconfigd` process to dump core, then that will be indicated. This could be caused by a bug in `vxconfigd`, particularly if *signal_name* is "`Segmentation fault`." Alternately, this could have been caused by a user sending `vxconfigd` a signal with the `kill` utility.

- **Action**

  Contact Customer Support.

A.2.2.2 Unrecognized operating mode

```
vxvm:vxconfigd: ERROR: mode_name: Unrecognized operating mode
```

- **Clarification**

  An invalid string was specified as an argument to the `-m` option. Valid strings are: `enable`, `disable`, and `boot`.

- **Action**

  Supply a correct option argument.

A.2.2.3 vxconfigd cannot boot-start RAID-5 volumes

```
vxvm:vxconfigd: ERROR: volume_name: vxconfigd cannot boot-start
RAID-5 volumes
```

- **Clarification**

  A volume that `vxconfigd` should start immediately upon booting the system (that is, the volume for the `/usr` file system) has a RAID-5 layout. The `/usr` file system should never be defined on a RAID-5 volume.

- **Action**

  It is likely that the only recovery for this is to boot the Volume Manager from a network-mounted root file system (or from a CD-ROM), and reconfigure the `/usr` file system to be defined on a regular non-RAID-5 volume.

A.2.2.4 Cannot get all disk groups from the kernel

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

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.5 Cannot get all disks from the kernel

```
vxvm:vxconfigd: ERROR: Cannot get all disks from the kernel: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.6 Cannot get kernel transaction state

```
vxvm:vxconfigd: ERROR: Cannot get kernel transaction state: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.7 Cannot get private storage from kernel

```
vxvm:vxconfigd: ERROR: Cannot get private storage from kernel:
reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.8 Cannot get private storage size from kernel

```
vxvm:vxconfigd: ERROR: Cannot get private storage size from
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.9 Cannot get record from the kernel

```
vxvm:vxconfigd: ERROR: Cannot get record record_name from the
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.10 Cannot kill existing daemon, pid=process-ID

```
vxvm:vxconfigd: ERROR: Cannot kill existing daemon, pid=process-ID
```

- **Clarification**

  The `-k` (kill existing `vxconfigd` process) option was specified, but a running configuration daemon process could not be killed. A configuration daemon process, for purposes of this discussion, is any process that opens the `/dev/vx/config`

device (only one process can open that device at a time). If there is a configuration daemon process already running, then the `-k` option causes a `SIGKILL` signal to be send to that process. If, within a certain period of time, there is still a running configuration daemon process, then the above error message will be displayed.

- **Action**

This error can result from a kernel error that has made the configuration daemon process unkillable, from some other kind of kernel error, or from some other user starting another configuration daemon process after the `SIGKILL` signal. This last condition can be tested for by running `vxconfigd -k` again. If the error message is displayed again, contact Customer Support.

A.2.2.11 Cannot make directory

```
vxvm:vxconfigd: ERROR: Cannot make directory directory_path: reason
```

- **Clarification**

`vxconfigd` failed to create a directory that it expects to be able to create. Directories that `vxconfigd` might try to create are `/dev/vx/dsk`, `/dev/vx/rdsk`, and `/var/vxvm/tempdb`. Also, for each disk group, `/dev/vx/dsk/`*diskgroup* and `/dev/vx/rdsk/`*diskgroup* directories are created. The system error related to the failure is given in *reason*. A system error of "`No such file or directory`" indicates that one of the prefix directories (for example, `/var/vxvm`) does not exist.

This type of error normally indicates that the VxVM packages were installed incorrectly. Such an error can also occur if alternate file or directory locations are specified on the command line, using the `-x` option. The `_VXVM_ROOT_DIR` environment variable may also relocate to a directory that lacks a `var/vxvm` subdirectory.

- **Action**

Try to create the directory manually and then issue the command `vxdctl enable`. If the error is due to incorrect installation of the VxVM packages, try to add the VxVM packages again.

A.2.2.12 Cannot open /etc/vfstab

```
vxvm:vxconfigd: ERROR: Cannot open /etc/vfstab: reason
```

- **Clarification**

  `vxconfigd` could not open the `/etc/vfstab` file, for the reason given. The `/etc/vfstab` file is used to determine which volume (if any) to use for the `/usr` file system. If the `/etc/vfstab` file cannot be opened, `vxconfigd` prints the above error message and exits.

- **Action**

  This error indicates that your root file system is currently unusable. You may be able to repair your root file system by mounting the root file system after booting from a network or CD-ROM root file system. If the root file system is defined on a volume, then see the procedures defined for recovering from a failed root file system in Appendix B, “Recovery“.

A.2.2.13 Cannot recover operation in progress

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

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.14 Cannot reset VxVM kernel

```
vxvm:vxconfigd: ERROR: Cannot reset VxVM kernel: reason
```

- **Clarification**

  The `-r reset` option was specified to `vxconfigd`, but the VxVM kernel drivers could not be reset. The most common reason for this is "`A virtual disk device is open`." That error indicates that a VxVM tracing device or volume device is open.

- **Action**

  If, for some reason, you really want to reset the kernel devices, you will need to track down and kill all processes that have a volume or VxVM tracing device open. Also, if any volumes are mounted as file systems, unmount those file systems.

  An error reason other than "`A virtual disk device is open`" should not normally occur unless there is a bug in the operating system or in the Volume Manager.

A.2.2.15 Cannot start volume, no valid complete plexes

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

- **Clarification**

  This error indicates that the volume for either the root or `/usr` file system cannot be started because the volume contains no valid plexes. This can happen, for example, if disk failures have caused all plexes to be unusable. It can also happen as a result of Actions that caused all plexes to become unusable (for example, forcing the dissociation of subdisks or detaching, dissociation, or offlining of plexes).

- **Action**

  This error may result from a drive that failed to spin up. If so, rebooting may fix the problem. If that does not fix the problem, then the only recourse is to restore the root or `/usr` file system or to reinstall the system. Restoring the root or `/usr` file system requires that you have a valid backup. See Appendix B, "Recovery" for information on how to fix problems with root or `/usr` file system volumes.

A.2.2.16 Cannot start volume, no valid plexes

```
vxvm:vxconfigd: ERROR: Cannot start volume volume, no valid plexes
```

- **Clarification**

  This error indicates that the volume for either the root or `/usr` file system cannot be started because the volume contains no valid plexes. This can happen, for example, if disk failures have caused all plexes to be unusable. It can also happen as a result of Actions that caused all plexes to become unusable (for example, forcing the dissociation of subdisks or detaching, dissociating, or offlining plexes).

- **Action**

  This error may result from a drive that failed to spin up. If so, rebooting may fix the problem. If that does not fix the problem, then the only recourse is to restore the root or `/usr` file system or to reinstall the system. Restoring the root or `/usr` file system requires that you have a valid backup. See Appendix B, “Recovery” for information on how to fix problems with root or `/usr` file system volumes.

A.2.2.17 Cannot start volume, volume state is invalid

```
vxvm:vxconfigd: ERROR: Cannot start volume volume, volume state is
invalid
```

- **Clarification**

  The volume for the root or `/usr` file system is in an unexpected state (not `ACTIVE`, `CLEAN`, `SYNC` or `NEEDSYNC`). This should not happen unless the system administrator circumvents the mechanisms used by the Volume Manager to create these volumes.

- **Action**

  The only recourse is to bring up the Volume Manager on a CD-ROM or NFS-mounted root file system and to fix the state of the volume. See Appendix B, “Recovery” for further information.

A.2.2.18 Cannot store private storage into the kernel

```
vxvm:vxconfigd: ERROR: Cannot store private storage into the
kernel: error
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.19 Differing version of vxconfigd installed

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

- **Clarification**

  A `vxconfigd` daemon was started after the stopping of an earlier `vxconfigd` with a non-matching version number. This can happen, for example, if you upgrade from an earlier release of VxVM and run `vxconfigd` without a reboot.

- **Action**

  To fix, reboot the system.

A.2.2.20 Disk, group, device: not updated with new host ID

```
vxvm:vxconfigd: ERROR: Disk disk, group group, device device: not
updated with new host ID
Error: reason
```

- **Clarification**

  This can result from using `vxdctl hostid` to change the VxVM host ID for the system. The error indicates that one of the disks in a disk group could not be updated with the new host ID. Most likely, this indicates that the given disk has become inaccessible or has failed in some other way.

- **Action**

  Try running the following to determine whether the disk is still operational:

  ```
  vxdisk check device
  ```

  If the disk is no longer operational, `vxdisk` should print a message such as:

  device`: Error: Disk write failure`

  This will result in the disk being taken out of active use in its disk group, if it has not been taken out of use already. If the disk is still operational (which should not be the case), `vxdisk` will print:

  device`: Okay`

  If the disk is listed as `Okay`, try `vxdctl hostid` again. If it still results in an error, contact Customer Support.

A.2.2.21 Disk group, Disk: Cannot auto-import group

```
vxvm:vxconfigd: ERROR: Disk group group, Disk disk: Cannot auto-
import group: reason
```

- **Clarification**

  On system startup, `vxconfigd` failed to import the disk group associated with the named disk. A message related to the specific failure is given in *reason*. Additional error messages may be displayed that give more information on the specific error. In particular, this is often followed by:

  ```
  vxvm:vxconfigd: ERROR: Disk group group: Errors in some
  configuration copies: Disk device, copy number: Block bno: error ...
  ```

  The most common reason for auto-import failures is excessive numbers of disk failures, making it impossible for the Volume Manager to find correct copies of the disk group configuration database and kernel update log. Disk groups usually have enough copies of this configuration information to make such import failures unlikely.

A more serious failure is indicated by error types of:

```
Format error in configuration copy
Invalid magic number
Invalid block number
Duplicate record in configuration
Configuration records are inconsistent
```

These errors indicate that all configuration copies have become corrupt (due to disk failures, writing on the disk by an application or the administrator, or bugs in the Volume Manager).

Some correctable errors may be indicated by other error messages that are displayed in conjunction with the auto-import failure message. Look up those other errors for more information on their cause.

Failure of an auto-import indicates that the volumes in that disk group will not be available for use. If there are file systems on those volumes, then the system may yield further errors resulting from inability to access the volume when mounting the file system.

- **Action**

If the error is clearly caused by excessive disk failures, then you may have to re-create the disk group and restore contents of any volumes from a backup. There may be other error messages that are displayed that provide further information. See those other error messages for more information on how to proceed. If those errors do not make it clear how to proceed, contact Customer Support.

A.2.2.22 Disk group, Disk: Group name collides with record in rootdg

```
vxvm:vxconfigd: ERROR: Disk group group, Disk device: Group name
collides with record in rootdg
```

- **Clarification**

The name of a disk group that is being imported conflicts with the name of a record in the `rootdg` disk group. VxVM does not allow this kind of conflict because of the way the `/dev/vx/dsk` directory is organized: devices corresponding to records in the root disk group share this directory with subdirectories for each disk group.

- **Action**

  Either remove or rename the conflicting record in the root disk group, or rename the disk group on import. See the `vxdg(1M)` manual page for information on how to use the `import` operation to rename a disk group.

A.2.2.23 Disk group, Disk: Skip disk group with duplicate name

```
vxvm:vxconfigd: ERROR: Disk group group, Disk device: Skip disk
group with duplicate name
```

- **Clarification**

  Two disk groups with the same name are tagged for auto-importing by the same host. Disk groups are identified both by a simple name and by a long unique identifier (disk group ID) assigned when the disk group is created. Thus, this error indicates that two disks indicate the same disk group name but a different disk group ID.

  The Volume Manager does not allow you to create a disk group or import a disk group from another machine, if that would cause a collision with a disk group that is already imported. Therefore, this error is unlikely to occur under normal use. However, this error can occur in the following two cases:

  - A disk group cannot be auto-imported due to some temporary failure. If you create a new disk group with the same name as the failed disk group and reboot, then the new disk group will be imported first, and the auto-import of the older disk group will fail with `group with duplicate name` (more recently modified disk groups have precedence over older disk groups).
  - A disk group is deported from one host using the `-h` option to cause the disk group to be auto-imported on reboot from another host. If the second host was already auto-importing a disk group with the same name, then reboot of that host will yield this error.

- **Action**

  If you want to import both disk groups, then rename the second disk group on import. See the `vxdg(1M)` manual page for information on how to use the `import` operation to rename a disk group.

A.2.2.24 Disk group: Cannot recover temp database

```
vxvm:vxconfigd: ERROR: Disk group group: Cannot recover temp
database: reason
Consider use of "vxconfigd -x cleartempdir" [see vxconfigd(1M)].
```

- **Clarification**

  This can happen if you kill and restart `vxconfigd` or you if you disable and enable it with `vxdctl disable` and `vxdctl enable`. This error indicates a failure related to reading the `/var/vxvm/tempdb/`*groupname* file. This is a temporary file used to store information that is used when recovering the state of an earlier `vxconfigd`. The file is re-created on a reboot, so this error should never survive a reboot.

- **Action**

  If you can reboot, do so. If you do not want to reboot, then do the following:

1. **Ensure that no `vxvol`, `vxplex`, or `vxsd` processes are running.**

   Use `ps -e` to search for such processes, and use `kill` to kill any that you find. You may have to run `kill` twice to make these processes go away. Killing utilities in this way may make it difficult to make administrative changes to some volumes until the system is rebooted.

2. **Type the command**:

   ```
   vxconfigd -x cleartempdir 2> /dev/console
   ```

   This will re-create the temporary database files for all imported disk groups.

   The `vxvol`, `vxplex`, and `vxsd` commands make use of these `tempdb` files to communicate locking information. If the file is cleared, then locking information can be lost. Without this locking information, two utilities can end up making incompatible changes to the configuration of a volume.

A.2.2.25 Disk group: Disabled by errors

```
vxvm:vxconfigd: ERROR: Disk group group: Disabled by errors
```

- **Clarification**

  This message indicates that some error condition has made it impossible for VxVM to continue to manage changes to a disk group. The major reason for this is that too many disks have failed, making it impossible for `vxconfigd` to continue to update configuration copies. There should be a preceding error message that indicates the specific error that was encountered.

  If the disk group that was disabled is the `rootdg` disk group, then the following additional error should be displayed:

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

  This additional message indicates that `vxconfigd` has entered the `disabled` state, which makes it impossible to change the configuration of any disk group, not just `rootdg`.

- **Action**

  If the underlying error resulted from a transient failure, such as a disk cabling error, then you may be able to repair the situation by rebooting. Otherwise, the disk group may have to be re-created and restored from a backup. Failure of the `rootdg` disk group may require reinstallation of the system if your system uses a root or `/usr` file system defined on a volume.

A.2.2.26 Disk group: Errors in some configuration copies: Disk, copy

```
vxvm:vxconfigd: ERROR: Disk group group: Errors in some
configuration copies: Disk disk, copy number: [Block number]:
reason ...
```

- **Clarification**

  During a failed disk group import, some of the configuration copies in the named disk group were found to have format or other types of errors which make those copies unusable. This message lists all configuration copies that have uncorrected errors, including any appropriate logical block number. If no other reasons are displayed, then this may be the cause of the disk group import failure.

- **Action**

  If some of the copies failed due to transient errors (such as cable failures), then a reboot or reimport may succeed in importing the disk group. Otherwise, the disk group may have to be re-created from scratch.

A.2.2.27 Disk group: Reimport of disk group failed

```
vxvm:vxconfigd: ERROR: Disk group group: Reimport of disk group
failed: reason
```

- **Clarification**

  After `vxconfigd` was stopped and restarted (or disabled and then enabled), the Volume Manager failed to re-create the import of the indicated disk group. The reason for failure is specified. Additional error messages may be displayed that give further information describing the problem.

- **Action**

  A major cause for this kind of failure is disk failures that were not addressed before `vxconfigd` was stopped or disabled. If the problem is a transient disk failure, then rebooting may take care of the condition.

A.2.2.28 Disk group: update failed

```
vxvm:vxconfigd: ERROR: Disk group group: update failed: reason
```

- **Clarification**

  I/O failures have prevented `vxconfigd` from updating any active copies of the disk group configuration. This usually indicates a large number of disk failures. This error will usually be followed by the message:

```
vxvm:vxconfigd: ERROR: Disk group group: Disabled by errors
```

- **Action**

  If the underlying error resulted from a transient failure, such as a disk cabling error, then you may be able to repair the situation by rebooting. Otherwise, the disk group may have to be re-created and restored from a backup.

A.2.2.29 Failed to store commit status list into kernel

```
vxvm:vxconfigd: ERROR: Failed to store commit status list into
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.30 GET_VOLINFO ioctl failed

```
vxvm:vxconfigd: ERROR: GET_VOLINFO ioctl failed: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.31 Get of current rootdg failed

```
vxvm:vxconfigd: ERROR: Get of current rootdg failed: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.32 Memory allocation failure

```
vxvm:vxconfigd: ERROR: Memory allocation failure
```

- **Clarification**

  This indicates that there is insufficient memory to start up the Volume Manager and to get the volumes for the root and `/usr` file systems running.

- **Action**

  This error should not normally occur, unless your system has very small amounts of memory. Adding just swap space will probably not help because this error is most likely to occur early in the boot sequence, before swap areas have been added.

A.2.2.33 Mount point: volume not in rootdg disk group

```
vxvm:vxconfigd: ERROR: Mount point path: volume not in rootdg disk
group
```

- **Clarification**

The volume device listed in the `/etc/vfstab` file for the given mount-point directory (normally `/usr`) is listed as in a disk group other than `rootdg`. This error should not occur if the standard VxVM procedures are used for encapsulating the disk containing the `/usr` file system.

- **Action**

1. **Boot the Volume Manager from a network or CD-ROM mounted root file system.**
2. **Start the Volume Manager using `fixmountroot` on a valid mirror disk of the root file system.**
3. **Mount the root file system volume and edit the `/etc/vfstab` file.**

   Change the file to use a direct partition for the file system. There should be a comment in the `/etc/vfstab` file that indicates which partition to use, for example:

```
#NOTE: volume usr (/usr) encapsulated partition c0t3d0s5
```

A.2.2.34 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:
    device type=device_type info=devinfo ...
Disks in alternate version of rootdg:
    device type=device_type info=devinfo ...
```

- **Clarification**

This message may occur when `vxconfigd` is not running in autoconfigure mode (see the `vxconfigd(1M)` manual page) and when, after several retries, it cannot 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/volboot` 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 is displayed and the root disk group importation fails.

- **Action**

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

A.2.2.35 Open of directory failed

```
vxvm:vxconfigd: ERROR: Open of directory directory failed: reason
```

- **Clarification**

An open failed for the `/dev/vx/dsk` or `/dev/vx/rdsk` directory (or a subdirectory of either of those directories). The only likely cause of such a failure should be that the directory was removed by the administrator or by an errant program. For this case, the *reason* should be "`No such file or directory.`" An alternate possible cause is an I/O failure.

- **Action**

If the error was "`No such file or directory`," then create the directory (using `mkdir`). Then run the command `vxdctl enable`.

If the error was an I/O error, then there may be other serious damage to the root file system. You may need to reformat your root disk and restore the root file system from backup. Contact your system vendor or consult your system documentation.

A.2.2.36 Read of directory failed

```
vxvm:vxconfigd: ERROR: Read of directory directory failed: reason
```

- **Clarification**

There was a failure in reading the `/dev/vx/dsk` or `/dev/vx/rdsk` directory (or a subdirectory of either of those directories). The only likely cause of this error is an I/O failure on the root file system.

- **Action**

  If the error was an I/O error, then there may be other serious damage to the root file system. You may need to reformat your root disk and restore the root file system from backup. Contact your system vendor or consult your system documentation.

A.2.2.37 System boot disk does not have a valid root plex

```
vxvm:vxconfigd: ERROR: System boot disk does not have a valid root
plex
Please boot from one of the following disks:
Disk: diskname Device: device ...
```

- **Clarification**

  The system is configured to use a volume for the root file system, but was not booted on a disk containing a valid mirror of the root volume. Disks containing valid root mirrors are listed as part of the error message. A disk is usable as a boot disk if there is a root mirror on that disk which is not stale or offline.

- **Action**

  Try to boot from one of the disks named in the error message.

  Under Solaris, you may be able to boot using a device alias for one of the named disks. For example, try:

  `boot vx-`*`diskname`*

A.2.2.38 System startup failed

```
vxvm:vxconfigd: ERROR: System startup failed
```

- **Clarification**

  Either the root or the `/usr` file system volume could not be started, rendering the system unusable. The error that resulted in this condition should appear prior to this error message.

- **Action**

  Look up other error messages appearing on the console and take the actions suggested in the descriptions of those messages.

A.2.2.39 There is no volume configured for the root device

```
vxvm:vxconfigd: ERROR: There is no volume configured for the root
device
```

- **Clarification**

  The system is configured to boot from a root file system defined on a volume, but there is no root volume listed in the configuration of the `rootdg` disk group.

  There are two possible causes of this error:

  1. The `/etc/system` file was erroneously updated to indicate that the root device is `/pseudo/vxio@0:0`. This should happen only as a result of direct manipulation by the administrator.

  2. The system somehow has a duplicate `rootdg` disk group, one of which contains a root file system volume and one of which does not, and `vxconfigd` somehow chose the wrong one. Since `vxconfigd` chooses the more recently accessed version of `rootdg`, this error can happen if the system clock was updated incorrectly at some point (causing the apparent access order of the two disk groups to be reversed). This can also happen if some disk group was deported and renamed to `rootdg` with locks given to this host.

- **Action**

  In case 1, boot the system on a CD-ROM or networking-mounted root file system, directly mount the disk partition of the root file system, and remove the following lines from `/etc/system`:

  ```
  rootdev:/pseudo/vxio@0:0
  set vxio:vol_rootdev_is_volume=1
  ```

  In case 2, either boot with all drives in the offending version of `rootdg` turned off, or import and rename [see `vxdg(1M)`] the offending `rootdg` disk group from another host. In the case of turning off drives, type the following command after booting:

  ```
  vxdg flush rootdg
  ```

  This will update time stamps on the imported version of `rootdg`, which should make the correct version appear to be the more recently accessed. If this does not correct the problem, contact Customer Support.

A.2.2.40 Unexpected configuration tid for group found in kernel

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

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.41 Unexpected error during volume reconfiguration

```
vxvm:vxconfigd: ERROR: Unexpected error during volume volume
reconfiguration: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.42 Unexpected error fetching disk for volume

```
vxvm:vxconfigd: ERROR: Unexpected error fetching disk for disk
volume: reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.43 Unexpected values stored in the kernel

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

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.2.44 Version number of kernel does not match vxconfigd

```
vxvm:vxconfigd: ERROR: Version number of kernel does not match
vxconfigd
```

- **Clarification**

  The release of `vxconfigd` does not match the release of the Volume Manager kernel drivers. This should happen only as a result of upgrading VxVM, and then running `vxconfigd` without a reboot.

- **Action**

  Reboot the system. If that does not cure the problem, then add the VxVM packages again.

A.2.2.45 Volume for mount point /usr not found in rootdg disk group

```
vxvm:vxconfigd: ERROR: Volume volume for mount point /usr not found
in rootdg disk group
```

- **Clarification**

  The system is configured to boot with `/usr` mounted on a volume, but the volume associated with `/usr` is not listed in the configuration of the `rootdg` disk group. There are a couple of possible causes of this error:

1. The `/etc/vfstab` file was erroneously updated to indicate the device for the `/usr` file system is a volume, but the volume named is not in the `rootdg` disk group. This should happen only as a result of direct manipulation by the administrator.

2. The system somehow has a duplicate `rootdg` disk group, one of which contains the `/usr` file system volume and one of which does not (or uses a different volume name), and `vxconfigd` somehow chose the wrong `rootdg`. Since `vxconfigd` chooses the more recently accessed version of `rootdg`, this error can happen if the system clock was updated incorrectly at some point (causing the apparent access order of the two disk groups to be reversed). This can also happen if some disk group was deported and renamed to `rootdg` with locks given to this host.

- **Action**

In case 1, boot the system on a CD-ROM or networking-mounted root file system. If the root file system is defined on a volume, then start and mount the root volume using the procedures defined in Appendix B, "Recovery". If the root file system is not defined on a volume, then just mount the root file system directly. Edit the `/etc/vfstab` file to correct the entry for the `/usr` file system.

In case 2, either boot with all drives in the offending version of `rootdg` turned off, or import and rename [see `vxdg(1M)`] the offending `rootdg` disk group from another host. In the case of turning off drives, run the following command after booting:

```
vxdg flush rootdg
```

This updates time stamps on the imported version of `rootdg`, which should make the correct version appear to be the more recently accessed. If this does not correct the problem, contact Customer Support.

A.2.2.46 cannot open /dev/vx/config

```
vxvm:vxconfigd: ERROR: cannot open /dev/vx/config: reason
```

- **Clarification**

The `/dev/vx/config` device could not be opened. `vxconfigd` uses this device to communicate with the Volume Manager kernel drivers. The *reason* string indicates the reason for the open failure. The most likely reason is `Device is already open`. This reason indicates that some process (most likely `vxconfigd`) already has

/dev/vx/config open. Other less likely reasons are "No such file or directory" or "No such device or address." For either of these two reasons, the two likely causes are:

- The Volume Manager package installation did not complete correctly.
- The device node was removed by the administrator or by an errant shell script.

- **Action**

For the reason "Device is already open," if you really want to run vxconfigd, then stop or kill the old one. You can kill whatever process has vxconfigd open by using the command:

```
vxdctl -k stop
```

For other failure reasons, consider re-adding the base Volume Manager package. This will reconfigure the device node and re-install the Volume Manager kernel device drivers. See the *Sun StorEdge Volume Manager 2.6 Installation Guide* for information on how to add the package using pkgadd. If you cannot re-add the package, then contact Customer Support for more information.

A.2.2.47 enable failed

```
vxvm:vxconfigd: ERROR: enable failed: reason
```

- **Clarification**

Regular startup of vxconfigd failed for the stated reason. This error can also result from the command vxdctl enable. This error may include the following additional text:

additional-reason; aborting
This message indicates that the failure was fatal and that vxconfigd is forced to exit. The most likely cause that results in an abort is inability to create IPC channels for communicating with other utilities.

additional-reason; transactions are disabled
This message indicates that vxconfigd is continuing to run, but no configuration updates are possible until the error condition is repaired.

Additionally, this may be followed with:

vxvm:vxconfigd: ERROR: Disk group group: Errors in some configuration copies:
Disk *device*, copy *number*: Block *bno*: *error* ...

Reasons for failure vary considerably. Other error messages may be displayed that further indicate the underlying problem. If the `Errors in some configuration copies` error occurs, then that may indicate the problem.

- **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.2.2.48 `/dev/vx/info`

```
vxvm:vxconfigd: ERROR: /dev/vx/info: reason
```

- **Clarification**

The `/dev/vx/info` device could not be opened, or did not respond to a Volume Manager kernel request. This error most likely indicates one of the following:

- The Volume Manager package installation did not complete correctly.
- The device node was removed by the administrator or by an errant shell script.

- **Action**

Consider re-adding the base Volume Manager package. This reconfigures the device node and re-installs the Volume Manager kernel device drivers. See the *Sun StorEdge Volume Manager 2.6 Installation Guide* for information on how to add the package using `pkgadd`.

A.2.3 `vxconfigd` Fatal Error Messages

The following are fatal error messages associated with `vxconfigd`.

A.2.3.1 `Disk group rootdg: Inconsistency -- Not loaded into kernel`

```
vxvm:vxconfigd: FATAL ERROR: Disk group rootdg: Inconsistency --
Not loaded into kernel
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.3.2 Group group: Cannot update kernel

```
vxvm:vxconfigd: FATAL ERROR: Group group: Cannot update kernel
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.3.3 Interprocess communication failure

```
vxvm:vxconfigd: FATAL ERROR: Interprocess communication failure:
reason
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.3.4 Invalid status stored in kernel

```
vxvm:vxconfigd: FATAL ERROR: Invalid status stored in kernel
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.3.5 Memory allocation failure during startup

```
vxvm:vxconfigd: FATAL ERROR: Memory allocation failure during
startup
```

- **Clarification**

  This indicates that there is insufficient memory to start up the Volume Manager and to get the volumes for the root and `/usr` file systems running.

- **Action**

  This error should not normally occur, unless your system has very small amounts of memory. Adding just swap space probably will not help, because this error is most likely to occur early in the boot sequence, before swap areas have been added.

A.2.3.6 Rootdg cannot be imported during boot

```
vxvm:vxconfigd: FATAL ERROR: Rootdg cannot be imported during
boot
```

- **Clarification**

  This is an internal Volume Manager error. This error should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.3.7 Unexpected threads failure

```
vxvm:vxconfigd: FATAL ERROR: Unexpected threads failure: reason
```

- **Clarification**

  This is an unexpected operating system error. This error should not occur unless there is a bug in the Volume Manager or in the Solaris multi-threading libraries.

- **Action**

  Contact Customer Support for more information.

A.2.4 vxconfigd Notice Messages

The following are notice messages associated with `vxconfigd`.

A.2.4.1 Detached disk

```
vxvm:vxconfigd: NOTICE: Detached disk disk
```

- **Clarification**

  The named disk appears to have become unusable and was detached from its disk group. Additional messages may appear to indicate other records detached as a result of the disk detach.

- **Action**

  If hot-relocation is enabled, the VxVM objects affected by the disk failure may be taken care of automatically. Mail will be sent to `root` indicating what actions were taken by the Volume Manager and what further actions the administrator should take.

A.2.4.2 Detached log for volume

```
vxvm:vxconfigd: NOTICE: Detached log for volume volume
```

- **Clarification**

  The DRL or RAID-5 log for the named volume was detached as a result of a disk failure, or as a result of the administrator removing a disk with `vxdg -k rmdisk`. A failing disk is indicated by a `Detached disk` *disk* message.

- **Action**

  If the log is mirrored, hot-relocation may automatically relocate the failed log. Remove the failing logs using either `vxplex dis` or `vxsd dis`. Then, use `vxassist addlog` [see the `vxassist(1M)` manual page] to add a new log to the volume.

A.2.4.3 Detached plex in volume

```
vxvm:vxconfigd: NOTICE: Detached plex plex in volume volume
```

- **Clarification**

  The specified plex was disabled as a result of a disk failure, or as a result of the administrator removing a disk with `vxdg -k rmdisk.` A failing disk is indicated by a `Detached disk` *disk* message.

- **Action**

  If hot-relocation is enabled, the VxVM objects affected by the disk failure may be taken care of automatically. Mail will be sent to `root` indicating what actions were taken by the Volume Manager and what further actions the administrator should take.

A.2.4.4 Detached subdisk in volume

```
vxvm:vxconfigd: NOTICE: Detached subdisk subdisk in volume volume
```

- **Clarification**

  The specified subdisk was disabled as a result of a disk failure, or as a result of the administrator removing a disk with `vxdg -k rmdisk`. A failing disk is indicated by a `Detached disk` *disk* message.

- **Action**

  If hot-relocation is enabled, the VxVM objects affected by the disk failure may be taken care of automatically. Mail will be sent to `root` indicating what actions were taken by the Volume Manager and what further actions the administrator should take.

A.2.4.5 Detached volume

```
vxvm:vxconfigd: NOTICE: Detached volume volume
```

- **Clarification**

  The specified volume was detached as a result of a disk failure, or as a result of the administrator removing a disk with `vxdg -k rmdisk.` A failing disk is indicated by a `Detached disk` *disk* message. Unless the disk error is transient and can be fixed with a reboot, the contents of the volume should be considered lost.

- **Action**

  There is no action to be taken. Contact Customer Support for more information.

A.2.4.6 Offlining config copy on disk

```
vxvm:vxconfigd: NOTICE: Offlining config copy number on disk disk:
Reason: reason
```

- **Clarification**

  An I/O error caused the indicated configuration copy to be disabled. This is a notice only, and does not normally imply serious problems, unless this is the last active configuration copy in the disk group.

- **Action**

  You should consider replacing the indicated disk, since this error indicates that the disk has deteriorated to the point where write errors cannot be repaired automatically. This can also result from transient errors, such as cabling problems or power problems. Check for a cabling problem.

A.2.4.7 Volume entering degraded mode

```
vxvm:vxconfigd: NOTICE: Volume volume entering degraded mode
```

- **Clarification**

  The detach of a subdisk in the named RAID-5 volume has caused that volume to enter “degraded” mode. While in degraded mode, performance of the RAID-5 volume will be substantially reduced. More importantly, failure of another subdisk may leave the RAID-5 volume unusable. Also, if the RAID-5 volume does not have an active log, then failure of the system may leave the volume unusable.

- **Action**

  If hot-relocation is enabled, the VxVM objects affected by the disk failure may be taken care of automatically. Mail will be sent to `root` indicating what actions were taken by the Volume Manager and what further actions the administrator should take.

A.2.5 `vxconfigd` Warning Messages

The following are warning messages associated with `vxconfigd`.

A.2.5.1 Bad request: client, portal [REQUEST|DIAG], size

```
vxvm:vxconfigd: WARNING: Bad request number: client number, portal
[REQUEST|DIAG], size number
```

- **Clarification**

  This is a diagnostic message that indicates an invalid request generated by a utility that has connected to `vxconfigd`. This message indicates a bug in that connected utility.

- **Action**

  If you are actually developing a new utility, then this error indicates a bug in your code. Otherwise, this error indicates a bug in the Volume Manager. Contact Customer Support for more information.

A.2.5.2 Cannot change disk group record in kernel

```
vxvm:vxconfigd: WARNING: Cannot change disk group record in
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.3 Cannot create device

```
vxvm:vxconfigd: WARNING: Cannot create device device_path: reason
```

- **Clarification**

  `vxconfigd` cannot create a device node either under `/dev/vx/dsk` or under `/dev/vx/rdsk`. This should happen only if the root file system has run out of inodes.

- **Action**

  Try removing some files from the root file system. Then, regenerate the device node with the command:

```
vxdctl enable
```

A.2.5.4 Cannot exec /usr/bin/rm to remove directory

```
vxvm:vxconfigd: WARNING: Cannot exec /usr/bin/rm to remove
directory: reason
```

- **Clarification**

  The given directory could not be removed because the `/usr/bin/rm` utility could not be executed by `vxconfigd`. This is not a serious error. The only side effect of a directory not being removed is that the directory and its contents will continue to use space in the root file system. However, this does indicate that the `/usr` file system is not mounted, or that the `rm` utility is missing or is not in its usual location. This may be a serious problem for the general running of your system.

- **Action**

  If the `/usr` file system is not mounted, you will need to determine how to get it mounted. If the `rm` utility is missing or is not in the `/usr/bin` directory, you should restore it from somewhere.

A.2.5.5 Cannot fork to remove directory

```
vxvm:vxconfigd: WARNING: Cannot fork to remove directory directory:
reason
```

- **Clarification**

  The given directory could not be removed because `vxconfigd` could not fork in order to run the `rm` utility. This is not a serious error. The only side effect of a directory not being removed is that the directory and its contents will continue to use space in the root file system. The most likely cause for this error is that your system does not have enough memory or paging space to allow `vxconfigd` to fork.

- **Action**

  If your system is this low on memory or paging space, then your overall system performance is probably being substantially effected. Consider adding more memory or paging space.

A.2.5.6 Cannot issue internal transaction

```
vxvm:vxconfigd: WARNING: Cannot issue internal transaction: reason
```

- **Clarification**

  This problem usually occurs only if there is a Volume Manager bug. However, it may occur in cases where memory is low.

- **Action**

  Contact Customer Support for more information.

A.2.5.7 Cannot open log file

```
vxvm:vxconfigd: WARNING: Cannot open log file log_filename: reason
```

- **Clarification**

  The `vxconfigd` console output log file could not be opened for the given reason. A log file is opened if `-x log` is specified, or if a log file is specified with `-x logfile=`*file*. The default log file is `/var/vxvm/vxconfigd.log`. The most likely cause for failure is "`No such file or directory`," which indicates that the directory containing the log file does not exist.

- **Action**

  Create any needed directories, or use a different log file path name.

A.2.5.8 Detaching plex from volume

```
vxvm:vxconfigd: WARNING: Detaching plex plex from volume volume
```

- **Clarification**

  The given plex is being detached from the given volume as part of starting the volume. This error only happens for volumes that are started automatically by `vxconfigd` at system startup (that is, for the root and `/usr` file system volumes). The plex is being detached as a result of an I/O failure, a disk failure during startup or prior to the last system shutdown or crash, or a disk removal prior to the last system shutdown or crash.

- **Action**

  If you want to ensure that the root or `/usr` file system retains the same number of active mirrors, then remove the given plex and add a new mirror using the `vxassist mirror` operation. You might also consider replacing any bad disks before using `vxassist mirror`.

A.2.5.9 `Disk in group flagged as shared; Disk skipped`

```
vxvm:vxconfigd: WARNING: Disk disk in group group flagged as
shared; Disk skipped
```

- **Clarification**

  The given disk is listed as shared, but the running version of VxVM does not support shared disk groups. This message can usually be ignored.

- **Action**

  There is no action to take. If you want to use the disk on this system, then use `vxdiskadd` to add the disk for use by the local system. However, do not do that if the disk really is in a shared disk group that is in use by other systems that are sharing this disk.

A.2.5.10 `Disk in group locked by host Disk skipped`

```
vxvm:vxconfigd: WARNING: Disk disk in group group locked by host
hostid Disk skipped
```

- **Clarification**

  The given disk is listed as locked by the host with the listed Volume Manager hostid (usually the same as the system hostname). This message can usually be ignored.

- **Action**

  There is no action to take. If you want to use the disk on this system, then use `vxdiskadd` to add the disk for use by the local system. However, *do not* do that if the disk really is in a disk group that is in use by another system that is sharing this disk.

A.2.5.11 Disk in group: Disk device not found

```
vxvm:vxconfigd: WARNING: Disk disk in group group: Disk device not
found
```

- **Clarification**

  No physical disk can be found that matches the named disk in the given disk group. This is equivalent to failure of that disk. Physical disks are located by matching disk IDs stored in the Volume Manager header on a disk and disk IDs stored in the disk group configuration. The configuration contains the official list of disk IDs for all disks in a disk group (the IDs are contained in disk media configuration records). The physical disks are then scanned to match that list against the disk IDs stored in disk headers. This error message is displayed for any disk IDs in the configuration that are not located in the disk header of any physical disk.

  This may result from a transient failure (such as a poorly-attached cable, or from a disk that failed to spin up fast enough). Alternately, this may happen as a result of a disk being physically removed from the system, or from a disk that has become unusable due to a head crash or electronics failure.

  Any RAID-5 or DRL log plexes on this disk will be unusable; any RAID-5 subdisks or mirrored plexes containing subdisks on this disk will also be unusable. These disk failures (particularly multiple disk failures) may cause one or more volumes to become unusable.

- **Action**

  If hot-relocation is enabled, the VxVM objects affected by the disk failure may be taken care of automatically. Mail will be sent to `root` indicating what actions were taken by the Volume Manager and what further actions the administrator should take.

A.2.5.12 Disk in kernel is not a recognized type

```
vxvm:vxconfigd: WARNING: Disk disk in kernel is not a recognized
type
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.13 Disk names group, but group ID differs

```
vxvm:vxconfigd: WARNING: Disk disk names group group, 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.

- **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.2.5.14 Disk group is disabled, disks not updated with new host ID

```
vxvm:vxconfigd: WARNING: Disk group group is disabled, disks not
updated with new host ID
```

- **Clarification**

  As a result of failures, the named disk group has become disabled. Earlier error messages should indicate the cause of this. This warning message indicates that disks in that disk group were not updated with a new Volume Manager host ID.

  This warning message should result only from a `vxdctl hostid` operation.

- **Action**

  Typically, unless a disk group was disabled due to transient errors, there is no way to repair a disabled disk group. The disk group may have to be reconstructed from scratch. If the disk group was disabled due to a transient error (such as a cabling

problem), then a future reboot may not automatically import the named disk group, due to the change in Volume Manager host ID for the system. In that case, the disk group should be imported directly using `vxdg import` with the `-C` option.

A.2.5.15 Disk group: Disk group log may be too small

```
vxvm:vxconfigd: WARNING: Disk group group: 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 occurs only during disk group import; it can occur only 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.

- **Action**

If this situation does occur, then the disks in the group will have to be explicitly reinitialized with larger log areas (which would require data to be restored from backup). See the `vxdisk(1M)` manual page. To reinitialize all of the disks, they must be detached from the group with which they are associated and then reinitialized and re-added. The disk group should then be deported and re-imported for the changes to the log areas for the group to take effect.

A.2.5.16 Disk group: Errors in some configuration copies: Disk, copy

```
vxvm:vxconfigd: WARNING: Disk group group: Errors in some
configuration copies: Disk disk, copy number: [Block number]: reason
...
```

- **Clarification**

During a disk group import, some of the configuration copies in the named disk group were found to have format or other types of errors that make those copies unusable. This message lists all configuration copies that have uncorrected errors, including any appropriate logical block number.

- **Action**

  There are usually enough configuration copies for any disk group to ensure that these errors do not become a serious problem. No action is usually necessary.

A.2.5.17 Error in volboot file

```
vxvm:vxconfigd: WARNING: Error in volboot file: reason Entry: disk
device  disk_type disk_info
```

- **Clarification**

  The `/etc/vx/volboot` file includes an invalid disk entry. This error should occur only if the file was edited directly (for example, using the `vi` editor).

- **Action**

  This is just a warning message. You can remove the offending entry by typing:

```
vxdctl rm disk device
```

A.2.5.18 Failed to store commit status list into kernel

```
vxvm:vxconfigd: WARNING: Failed to store commit status list into
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.19 Failed to update voldinfo area in kernel

```
vxvm:vxconfigd: WARNING: Failed to update voldinfo area in
kernel: reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.20 Field too long in volboot file

```
vxvm:vxconfigd: WARNING: Field too long in volboot file:
Entry: disk device disk_type disk_info
```

- **Clarification**

  The `/etc/vx/volboot` file includes a disk entry with a field that is larger than the size the Volume Manager supports. This error should occur only if the file was edited directly (for example, using the `vi` editor).

- **Action**

  This is just a warning message. You can remove the offending entry by typing:

  ```
  vxdctl rm disk device
  ```

A.2.5.21 Get of record from kernel failed

```
vxvm:vxconfigd: WARNING: Get of record record_name from kernel
failed: reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.22 Group: Duplicate virtual device number(s)

```
vxvm:vxconfigd: WARNING: Group group: Duplicate virtual device
number(s):
Volume volume remapped from major,minor to major,minor ...
```

- **Clarification**

  The configuration of the named disk group includes conflicting device numbers. A disk group configuration lists the recommended device number to use for each volume in the disk group. If two volumes in two disk groups happen to list the same device number, then one of the volumes must use an alternate device number. This is called device number remapping. Remapping is a temporary change to a volume. If the other disk group is deported and the system is rebooted, then the volume that was remapped may no longer be remapped. Also, volumes that are remapped once are not guaranteed to be remapped to the same device number in further reboots.

- **Action**

  Use the `vxdg reminor` operation to renumber all volumes in the offending disk group permanently. See the `vxdg(1M)` manual page for more information.

A.2.5.23 Internal transaction failed

```
vxvm:vxconfigd: WARNING: Internal transaction failed: reason
```

- **Clarification**

  This problem usually occurs only if there is a Volume Manager bug. However, it may occur in cases where memory is low.

- **Action**

  Contact Customer Support for more information.

A.2.5.24 cannot remove group from kernel

```
vxvm:vxconfigd: WARNING: cannot remove group group from kernel:
reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.25 client not recognized by VXVM library

```
vxvm:vxconfigd: WARNING: client number not recognized by VXVM
library
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.26 client not recognized

```
vxvm:vxconfigd: WARNING: client number not recognized
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.27 library and vxconfigd disagree on existence of client

```
vxvm:vxconfigd: WARNING: library and vxconfigd disagree on
existence of client number
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.28 library specified non-existent client

```
vxvm:vxconfigd: WARNING: library specified non-existent client
number
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.29 response to client failed

```
vxvm:vxconfigd: WARNING: response to client number failed: reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.2.5.30 vold_turnclient failed

```
vxvm:vxconfigd: WARNING: vold_turnclient(number) failed reason
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.3 Kernel Error Messages

The following sections cover the kernel level error messages.

A.3.1 Kernel Notice Messages

The following are notice messages associated with the kernel.

A.3.1.1 Can't open disk in group

```
vxvm:vxio:NOTICE: Can't open disk disk in group disk_group. 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
number
```

- **Clarification**

  The named disk cannot be accessed in the named disk group.

- **Action**

  Ensure that the disk exists, is powered on, and is visible to the system.

A.3.1.2 Can't close disk in group

```
vxvm:vxio:NOTICE: Can't close disk disk in group disk_group. If it
is removable media (like a floppy), it may have been removed.
Otherwise, there may be problems with the drive. Kernel error code
public_region_error/private_region_error
```

- **Clarification**

  This is unlikely to happen; closes cannot fail.

- **Action**

  None.

A.3.1.3 Read error on object of mirror in volume corrected

```
vxvm:vxio:NOTICE: read error on object subdisk of mirror plex in
volume volume (start offset, length length) corrected
```

- **Clarification**

  A read error occurred, which caused a read of an alternate mirror and a writeback to the failing region. This writeback was successful and the data was corrected on disk.

- **Action**

  No action is required. The problem was corrected automatically. The administrator may, however, note the failure as a reference because if the same region fails again or frequently, the error could indicate a more insidious failure and the disk should be reformatted at the next reasonable opportunity.

A.3.1.4 *String* on volume device in disk group

```
vxvm:vxio:NOTICE: string on volume device_# (device_name) in disk
group group_name
```

- **Clarification**

  An application requested message. The application running on top of the Volume Manager has requested the output of this message.

- **Action**

  Refer to documentation for the appropriate application for more information.

A.3.1.5 Path Failure detected by vxdmp driver

```
vxvm:vxdmp:NOTICE: Path failure on <major>/<minor>
```

- **Clarification**

  Appears when a path under the control of the DMP driver fails. The device number of the failed path is part of the message.

- **Action**

  None.

A.3.1.6 Load of sd driver failed

```
vxvm:vxdmp:NOTICE: Could not load sd driver
```

- **Clarification**

  Appears when a path under the control of the DMP driver fails. The device number of the failed path is part of the message.

- **Action**

  None.

A.3.1.7 Install of sd driver failed

```
vxvm:vxdmp:NOTICE: Could not install sd driver
```

- **Clarification**

  During initialization, if the `vxdmp` driver tries to load the `sd` driver. If the attempt fails, this message appears.

- **Action**

  None.

A.3.1.8 Can’t lock sd driver

```
vxvm:vxdmp:NOTICE: could not lock sd driver
```

- **Clarification**

  The `sd` driver is locked during `vxdmp` driver initialization to avoid unloading of the driver. The message is printed when the `sd` driver cannot be locked.

- **Action**

  None.

A.3.1.9 Load of ssd driver failed

```
vxvm:vxdmp:NOTICE: Could not load ssd driver
```

- **Clarification**

  Appears when a path under the control of the DMP driver fails. The device number of the failed path is part of the message.

- **Action**

  None.

A.3.1.10 Install of ssd driver failed

```
vxvm:vxdmp:NOTICE: Could not install ssd driver
```

- **Clarification**

  During initialization, the `vxdmp` driver tries to load the `ssd` driver. If the attempt fails, this message appears.

- **Action**

  None.

A.3.1.11 Can’t lock ssd driver

```
vxvm:vxdmp:NOTICE: could not lock ssd driver
```

- **Clarification**

  The `ssd` driver is locked during `vxdmp` driver initialization to avoid unloading of the driver. The message is printed when the `ssd` driver cannot be locked.

- **Action**

  None.

A.3.2 Kernel Warning Messages

The following are warning messages associated with the kernel.

A.3.2.1 `Received spurious close`

```
vxvm:vxio:WARNING: Device major, minor: Received spurious close
```

- **Clarification**

  This message happens if a close was received for an object that was previously not opened. This will only happen if the operating system is not correctly tracking opens and closes.

- **Action**

  No action is necessary; the system will continue.

A.3.2.2 Failed to log the detach of the DRL volume

```
vxvm:vxio:WARNING: Failed to log the detach of the DRL volume
volume
```

- **Clarification**

  An attempt to write a kernel log entry indicating the loss of a DRL volume failed. The attempted write to the log failed either because the kernel log is full or because of a write error to the drive. The volume will become detached.

- **Action**

  Messages about log failures are often fatal, unless the problem is transient. However, the kernel log is sufficiently redundant that such errors are unlikely to occur.

  If the problem is not transient (that is, the drive cannot be fixed and brought back online without data loss), the disk group must be re-created from scratch and all of its volumes must be restored from backup. Even if the problem is transient, the system must be rebooted after correcting the problem.

  If error messages were seen from the disk driver, it is likely that the last copy of the log failed due to a disk error. The failed drive in the disk group should be replaced and the log will then be re-initialized on the new drive. The failed volume can then be forced into an active state and the data recovered. See Appendix B, “Recovery” for further information.

A.3.2.3 DRL volume is detached

```
vxvm:vxio:WARNING: DRL volume volume is detached
```

- **Clarification**

  A Dirty Region Logging volume became detached because a DRL log entry could not be written. This might be because of a media failure, in which case other errors may have been logged to the console.

- **Action**

  The volume containing the DRL log will continue. If the system fails before the DRL can be repaired, a full recovery of the volume’s contents may be necessary and will be performed automatically when the system is restarted. To recover the DRL capability, a new DRL log should be added to the volume using the `vxassist addlog` command.

A.3.2.4 Read error on mirror of volume

```
vxvm:vxio:WARNING: read error on mirror plex of volume volume offset
offset length length
```

- **Clarification**

  An error was detected while reading a mirror. This error may lead to further action shown by later error messages.

- **Action**

  If the volume is mirrored, no action is necessary at this time, since the alternate mirror's contents will be written to the failing mirror; this is often sufficient to correct media failures. If this error occurs often but never leads to a plex detach, there may be a marginal region on the disk at the position shown. You may eventually have to remove data from this disk (see the `vxevac(1M)` manual page) and then reformat the drive. In the unmirrored case, this message indicates that some data could not be read. The file system or other application reading the data may report an additional error, but in either event, data has been lost. The volume can be partially salvaged and moved to another location if desired.

A.3.2.5 Write error on mirror of volume offset length

```
vxvm:vxio:WARNING: write error on mirror plex of volume volume
offset offset length length
```

- **Clarification**

  An error was detected while writing a mirror. This error will generally be followed by a detach message, unless the volume is un-mirrored.

- **Action**

  The disk reporting the error is failing to correctly store written data. If the volume is not mirrored, consider removing the data and reformatting the disk. If the volume is mirrored, it will become detached and you should consider replacing or reformatting the disk.

  If this error occurs often but never leads to a plex detach, there may be a marginal region on the disk at the position shown. You may eventually have to remove data from this disk (see the `vxevac(1M)` manual page) and then reformat the drive.

A.3.2.6 Object detached from volume

```
vxvm:vxio:WARNING: object plex detached from volume volume
```

- **Clarification**

  An uncorrectable error was detected by the mirroring code and a mirror copy was detached.

- **Action**

  To restore redundancy, you may have to add another mirror. You should evacuate and reformat the disk on which the failure occurred, if possible. If the drive has failed completely, you may have to replace it.

A.3.2.7 Overlapping mirror detached from volume

```
vxvm:vxio:WARNING: Overlapping mirror plex detached from volume
volume
```

- **Clarification**

  An error has occurred on the last complete plex in the mirrored volume. Any sparse mirrors that map the failing region must be detached so that they cannot be accessed to satisfy that failed region inconsistently. This message indicates that such an overlapping mirror was found and is being detached.

- **Action**

  No Action is directly necessary. The message indicates that the volume may have left some data inaccessible at the failing region and that it is no longer redundantly stored.

A.3.2.8 Kernel log full

```
vxvm:vxio:WARNING: Kernel log full: volume detached
```

- **Clarification**

  A plex detach failed because the kernel log was full. As a result, the mirrored volume will become detached.

- **Action**

  It is unlikely that this condition could ever occur. The only corrective action for the detached volume is to reboot the system.

A.3.2.9 Kernel log update failed

```
vxvm:vxio:WARNING: Kernel log update failed: volume detached
```

- **Clarification**

  A plex detach failed because the kernel log could not be flushed to disk. As a result, the mirrored volume will become detached. This could be caused by all the disks containing a kernel log going bad.

- **Action**

  Correct the failed disks so that kernel logging can once again function.

A.3.2.10 Detaching RAID-5 volume

```
vxvm:vxio:WARNING: detaching RAID-5 raidvol
```

- **Clarification**

  Either a double-failure condition in the RAID-5 volume has been detected in the kernel or some other fatal error is preventing further use of the array.

- **Action**

  If two or more drives were lost due to a controller or power failure, then once the disks can be re-attached to the system, they should be recovered using the `vxrecover` utility. Check the console for other errors that may provide additional information as to the nature of the failure.

A.3.2.11 Object detached from RAID-5 volume

```
vxvm:vxio:WARNING: object subdisk detached from RAID-5 raidvol at
column column offset offset
```

- **Clarification**

A subdisk was detached from a RAID-5 volume at the specified column number and offset. This is caused by the failure of a disk or an uncorrectable error occurring on that disk.

- **Action**

Check the console for other error messages indicating the cause of the failure. If the disk has failed, replace it as soon as possible.

A.3.2.12 RAID-5 volume entering degraded mode operation

```
vxvm:vxio:WARNING: RAID-5 raidvol entering degraded mode operation
```

- **Clarification**

This message occurs when an uncorrectable error has forced the detach of a subdisk. At this point, not all data disks exist to provide the data upon request. Instead, parity regions are required to regenerate the data for each stripe in the array. Accesses will consequently take longer and will involve reading from all drives in the stripe.

- **Action**

Check the console for other error messages indicating the cause of the failure. If the disk has failed, replace it as soon as possible.

A.3.2.13 Double failure condition detected on RAID-5 volume

```
vxvm:vxio:WARNING: Double failure condition detected on RAID-5
raidvol
```

- **Clarification**

Double-failures occur if I/O errors are received at the same altitude in the array from more than one column of the array. This could be caused by a controller failure causing more than a single drive to become unavailable; by the loss of a second drive after having run in a degraded state for significant periods of time; or by two separate disk drives failing simultaneously (which is unlikely to happen).

- **Action**

  If the condition is correctable and the drives recoverable, the conditions should be corrected. The volume can then be recovered using the `vxrecover(1M)` command.

A.3.2.14 Failure in RAID-5 logging operation

```
vxvm:vxio:WARNING: Failure in RAID-5 logging operation
vxvm:vxio:WARNING: log object object_name detached from RAID-5
volume
```

- **Clarification**

  These two messages are displayed together when a RAID-5 log has failed and has been detached.

- **Action**

  To restore RAID-5 logging to the RAID-5 volume, simply create a new log region and attach it to the volume.

A.3.2.15 Stranded ilock on object

```
vxvm:vxio:WARNING: check_ilocks: stranded ilock on object_name
start offset len length
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.3.2.16 Overlapping ilocks

```
vxvm:vxio:WARNING: check_ilocks: overlapping ilocks: offset for
length, offset for length
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

A.3.2.17 Illegal vminor encountered

```
vxvm:vxio:WARNING: Illegal vminor encountered
```

- **Clarification**

  This message could result if a volume device other than the root volume device is opened before a configuration has been loaded.

- **Action**

  No action should be necessary; an attempt to access a volume device was made before the volume daemon (`vxconfigd`) loaded the volume configuration. Under normal startup conditions, this message should not occur. If the operation is necessary, start the Volume Manager and re-attempt the operation.

A.3.2.18 Uncorrectable read error

```
vxvm:vxio:WARNING: object_type object_name block offset: Uncorrectable
read error
```

- **Clarification**

  A read or write operation from the specified object failed. An error will be returned to the application.

- **Action**

  This error represents lost data. The data may need to be restored and failed media may need to be repaired. Depending on the type of object failing and on the type of recovery suggested for that type, an appropriate recovery operation may be necessary.

A.3.2.19 Uncorrectable read/write error

```
vxvm:vxio:WARNING: object_type object_name block offset:
Uncorrectable read error on object_type object_name block offset
vxvm:vxio:WARNING: object_type object_name block offset:
Uncorrectable write error on object_type object_name block offset
```

- **Clarification**

  A read or write operation from the specified object failed. An error will be returned to the application. Although similar to the previous message, this message is able to supply more specific information about the failing object.

- **Action**

  This error represents lost data. The data may need to be restored and failed media may need to be repaired. Depending on the type of object failing and on the type of recovery suggested for that type, an appropriate recovery operation may be necessary.

A.3.2.20 Root volumes are not supported on your PROM version

```
vxvm:vxio:WARNING: Root volumes are not supported on your PROM
version.
```

- **Clarification**

  The Volume Manager requires the ability to access the PROMs for your SPARC hardware. If the PROMs are not a recent OpenBoot™ PROM type, then root volumes will not be usable.

- **Action**

  If you have set up a root volume, then undo the configuration (by running `vxunroot` or removing the `rootdev` line from `/etc/system`) as soon as possible, and contact your hardware vendor for an upgrade to your PROM level.

A.3.2.21 `Cannot find device number`

```
vxvm:vxio:WARNING: Cannot find device number for boot_path
```

- **Clarification**

  The supplied boot path was retrieved from the PROMs for your system. It cannot be converted to a valid device number.

- **Action**

  Check your PROM settings for the correct boot string.

A.3.2.22 `mod_install returned` *errno*

```
vxvm:vxio:WARNING: mod_install returned errno
```

- **Clarification**

  A call made to the Solaris `mod_install()` function to load the `vxio` driver failed.

- **Action**

  Check your console for additional messages that may explain why the load failed. Also check the console messages log file for any additional messages that were logged but not displayed on the console.

A.3.2.23 subdisk failed in plex in volume

```
vxvm:vxio:WARNING: subdisk subdisk failed in plex plex in volume
volume
```

- **Clarification**

  The kernel has detected a subdisk failure, which may mean that the underlying disk is failing.

- **Action**

  Check for obvious problems with the disk (such as a disconnected cable). If hot-relocation is enabled and the disk is failing, the subdisk failure may be taken care of automatically.

A.3.3 Kernel Panic Messages

The following are panic messages associated with the kernel.

A.3.3.1 Object association depth overflow

```
vxvm:vxio:PANIC: Object association depth overflow
```

- **Clarification**

  This is an internal Volume Manager problem. This warning should not occur unless there is a bug in the Volume Manager.

- **Action**

  Contact Customer Support for more information.

APPENDIX **B**

Recovery

The Sun StorEdge Volume Manager provides the ability to protect systems from disk failures and to recover from disk failures. This appendix describes various recovery procedures and provides information to help you prevent loss of data or system access due to disk failures. It also describes possible plex and volume states.

For information specific to volume recovery, refer to Chapter 4, “Volume Administration”.

B.1 Protecting Your System

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 failure of a key disk (a disk involved with system operation). The Sun StorEdge Volume Manager provides the ability to protect your system from either type of problem.

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 following are some suggestions on how to protect your system and data:

- Place the disk containing the root file system (that is, the *root* or *boot* disk) under Volume Manager control (through encapsulation). This converts the `root` and `swap` devices to volumes (`rootvol` and `swapvol`). You should then mirror the root disk so that an alternate root disk exists for booting purposes. By mirroring disks critical to booting, you ensure that no single disk failure will leave your system unbootable and unusable.

  For maximum availability of the system, you should have mirrors for the `rootvol`, `swapvol`, `usr`, and `var` volumes. See Section B.3, “Possible Root (/), swap, and usr Configurations” for more information.

- Use mirroring to protect data. By mirroring your data, you prevent data loss from a disk failure. To preserve data, create and use mirrored volumes that have at least two data plexes. The plexes must be on different disks. If a disk failure causes a plex to fail, the data in the mirrored volume will still exist on the other disk.

  If you use `vxassist mirror` to create mirrors, it locates the mirrors so that the loss of one disk does not result in a loss of data. By default, `vxassist` does not create mirrored volumes; you can edit the `/etc/default/vxassist` file to set the default layout to mirrored. Refer to Chapter 4, "Volume Administration" for information on the `vxassist` defaults file.
- Leave the Volume Manager's hot-relocation feature on so that it can automatically detect failures, notify you of the nature of the failures, attempt to relocate any affected subdisks that are redundant, and initiate recovery procedures. Try to provide at least one hot-relocation spare disk per disk group to make sure that sufficient space is available for relocation if a failure occurs.

  If the root disk is mirrored, hot-relocation can automatically create another mirror of the root disk if the original root disk fails. The `rootdg` disk group should therefore contain enough contiguous spare or free space to accommodate the volumes on the root disk (`rootvol` and `swapvol` volumes require contiguous disk space).
- For mirrored volumes, take advantage of the Dirty Region Logging feature to speed up recovery of your mirrored volumes after a system crash. Make sure that each mirrored volume has at least one log subdisk. (Note that `rootvol`, `swapvol`, and `usr` volumes cannot be Dirty Region Logging volumes.)
- For RAID-5 volumes, take advantage of logging to prevent corruption of recovery data. Make sure that each RAID-5 volume has at least one log plex.
- Perform regular backups 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. Alternatively, a mistyped command could remove critical files or damage a file system directly.

B.2 The UNIX Boot Process

When turned on, a Sun SPARC™ machine will prompt for the `boot` command, unless the `autoboot` flag has been set in the non-volatile storage area used by the firmware. Machines with older PROMs have different prompts than the prompt for

V2 and V3 versions of PROM, which are relatively new. These newer versions of PROM are also known as OpenBoot PROMs (OBP). The `boot` command has a different syntax for these two types of PROMs:

```
ok boot [OBP names] [filename] [boot-flags]
```

OBP names specify the open boot PROM designations. For example, on desktop SPARC systems, the designation

`/sbus/esp@0,800000/sd@3,0:a`

indicates a SCSI disk (`sd`) at target 3, lun 0 on the SCSI bus, with the esp host adapter plugged into slot 0.

Note – With VxVM, it is also possible to boot using boot disk alias names. These aliases can take the form of VxVM provided names (such as `vx-rootdisk` or `vx-disk01`) or Solaris provided names (such as `disk1`). To view a list of possible bootable devices, enter `devalias` at the OpenBoot `ok` prompt.

The *filename* is the name of a standalone program to the `boot` program. The default is to boot `/kernel/unix` from the `root` partition, but you can specify another program (such as `/stand/diag`) on the command line. Some versions of the firmware allow the default file name to be saved in the non-volatile storage area of the system.

The `boot` program interprets the `-a` flag to mean “ask me” and prompts you for the name of the standalone program to boot. The `-a` flag is then passed to the standalone program.

Note – A system running VxVM with rootability will not boot with the defaults presented by the `-a` flag. See Section B.4.4.2, “/etc/system Copy Available” for the correct responses to `boot -a`.

The `boot` program passes all boot-flags to the file identified by *filename*. Boot-flags are not interpreted by the `boot` program. See the `kernel(1)` and `kadb(1M)` manual pages for information on the options available with the default standalone program, `/kernel/unix`.

B.2.1 Booting After Failures

If the root disk is mirrored, an alternate boot disk can be used to boot the system if the primary boot disk fails. To boot the system after failure of the primary boot disk:

1. **Check for aliased VM disks using the** `devalias` **command at the OpenBoot command prompt.**

   Disks that are suitable mirrors of the root disk will be listed with the name `vx-`*medianame*, where *medianame* represents the disk media name for the disk containing the candidate root file system.

2. **Type:**

```
ok boot alias_name
```

   where *alias_name* represents the aliased name of the selected disk.

   If a selected disk contains a root mirror that is stale, `vxconfigd` displays an error stating that the mirror is unusable and will list any non-stale alternate disks to boot from.

B.3 Possible Root (`/`), `swap`, and `usr` Configurations

While installing, different configurations are possible for root, `swap`, and `usr` file systems. The following cases are possible:

- `usr` is a directory under the root and no separate partition is allocated for it. In this case, `usr` becomes part of the `rootvol` volume when the root disk is encapsulated and put under VxVM control.
- `usr` is on a separate partition that is on the root disk. In this case, a separate volume will be created for the `usr` partition. `vxmirror` will mirror the `usr` volume on the destination disk.
- `usr` is on a separate partition that is not on the root disk. In this case, a volume will be created for the `usr` partition only if that disk is encapsulated by VxVM. Note that in such cases, encapsulating the root disk and having mirrors of the root volume will not help if the `usr` partition becomes inaccessible for any reason. You should therefore encapsulate both the disk containing the `usr` partition and the root disk, and have mirrors for the `usr`, `rootvol`, and `swapvol` volumes for maximum availability of the system.

The `rootvol` volume must exist in the `rootdg` disk group. Refer to Section 1.1.10.2, “Boot-Time Volume Restrictions” in Chapter 1 for information on `rootvol` and `usr` volume restrictions.

Solaris 2.x enables you to put `swap` partitions on any disk; it does not need an initial `swap` area during early phases of the boot process. By default, the Solaris 2.x installation chooses partition 0 on the selected root disk as the `root` partition and partition 1 as the `swap` partition. However, it is possible to have the `swap` partition on a partition not located on the root disk. In such cases, you are advised to encapsulate that disk and create mirrors for the `swap` volume. If you do not do this, damage to the `swap` partition will eventually cause the system to crash. It may be possible to boot the system, but having mirrors for the `swapvol` volume will prevent system failures.

B.3.1 Repairing Root (/) or /usr File Systems on Volumes

If the root (`/`) or `/usr` file system becomes unusable, it is desirable to be able to boot from a network-mounted root file system or from the Solaris installation CD-ROM, mount the file system, and then repair it. This is necessary, for example, if certain key files involved in booting have been corrupted.

This task is made more difficult when the root or `/usr` file system is defined on a mirrored volume, because changes to the partition that underlies one of the mirrors can result in corruption when the Volume Manager later boots and presumes that the mirrors are reasonably synchronized.

There are two workarounds for this:

- In many situations, the simplest workaround is to mount one plex of the root or `/usr` file system, repair it, unmount it, and use `dd` to copy the fixed plex to all other plexes. However, this can be error prone.
- Another workaround (which can also be used if one of the mirrors is striped, non-contiguous, or has no exactly-mapped underlying partition) is to use a set of scripts and files that are provided on the Volume Manager installation CD-ROM. If you can mount the CD-ROM or make its contents available over the network, you can run the Volume Manager.

Note – Not all versions of Solaris CD-ROM boot kernels support the use of the VxVM recovery scripts. If these scripts fail, the first workaround must be used.

Use the latter workaround to access the Volume Manager from CD-ROM and then perform any necessary repairs as follows:

1. **Make the contents of the scripts directory of the CD-ROM available while booted from your network or CD-ROM-mounted boot environment.**

   For example:

```
mkdir /tmp/cdrom
mount your_server:/cdrom/vrts_osm_2_3 /tmp/cdrom
```

2. **Set up a special environment for running the Volume Manager by typing:**

```
. /tmp/cdrom/scripts/fixsetup
```

   You must run this command from the Bourne shell or Korn shell. This script sets environment variables, so if you want to use the C shell, you must run `csh` as a subshell.

   The `fixsetup` command prompts for the location of the VxVM CD-ROM image. For this example, enter `/tmp/cdrom`.

3. **Try to mount the root file system read-only and get VxVM startup files by typing:**

```
fixmountroot
```

   When this command prompts for a disk containing a mirror of the root file system, enter a device name. For example, if you normally boot from disk 3 on your SCSI controller, respond with `c0t3d0`. If mounting of the root file system mirror is successful, `fixmountroot` will ask if you want to start up the Volume Manager. You

should normally respond with `yes`. It will then ask if you wish to start all volumes in the `rootdg` and other disk groups. You should normally start volumes in the `rootdg` disk group, but not in other disk groups.

The following is a sample `fixmountroot` interaction:

```
This script can be used to mount a root file system that has been
encapsulated as a volume. The file system will be mounted read-only,
using a file system slice that underlies the root file system volume.

Please enter the name of a disk containing a root volume mirror:  c0t3d0
!mount -o ro /dev/dsk/c0t3d0s0 /tmp/root_mirror

You can now choose to retrieve configuration files from the root volume
mirror, and to start up the Volume Manager.


Do you wish to start the Volume Manager using these files? [yes] yes
!cp /tmp/root_mirror/etc/vx/volboot /tmp/vxroot/etc/vx/volboot
!vxconfigd

Do you wish to start volumes in the rootdg disk group? [yes] yes
!vxrecover -svn -g rootdg
dg rootdg usetype fsgen: start linux linux-a2 linux-a3 nur_vol shv vm-build
dg rootdg usetype root: start rootvol
dg rootdg usetype swap: start vswap

Do you wish to start volumes in all other disk groups? [no] no

You can start all volumes in all known disk groups using the command:

        vxrecover -svn
```

The root file system mirror (that you specified to `fixmountroot`) is mounted read-only under the mount point `/tmp/root_mirror`. You can go to that directory and examine the root file system for errors, but you cannot fix the root file system directly under that mount point because it is a read-only mount.

4. **If you want to mount and repair the root file system (or the `/usr` file system), you must mount the volume as follows:**

```
mkdir /tmp/rootvol
fsck -F ufs -y /dev/vx/rdsk/rootvol
mount /dev/vx/dsk/rootvol /tmp/rootvol
```

You can use most Volume Manager commands, including the `vxdiskadm` menus, while booted in this environment. You can now edit or repair files such as `/etc/system`. When you are done with any repairs, unmount the volume and reboot.

If the `fixmountroot` command fails (for example, if the file system is too corrupted to be mounted) or some of the Volume Manager startup files are missing from the mounted root file system, you can start the Volume Manager directly by typing:

```
fixstartup
```

This command proceeds with a series of prompts that require you to enter the host identifier that VxVM uses for your host (normally the host name, not the host ID) and any license keys that are needed for correct operation. Respond to the prompts appropriately.

B.3.2 Backing Up and Restoring the Root File System

You should back up your root file system so that it can be restored if it is ever damaged.

If you are using the `ufs` file system, you can back up your root file system by typing:

```
/usr/lib/fs/ufs/ufsdump [dump-options] /dev/vx/rdsk/rootvol
```

You can then restore your root file system after a failure as follows:

1. **Boot from a CD-ROM or network-mounted root file system, then get the Volume Manager running (see Section B.3.1, “Repairing Root (/) or /usr File Systems on Volumes”).**

2. **Mount and restore the root file system by typing:**

```
newfs /dev/vx/rdsk/rootvol
mount /dev/vx/dsk/rootvol /mnt
cd /mnt
/usr/lib/fs/ufs/ufsrestore [restore-options]
```

B.4 Failures and Recovery Procedures

While there are many types of failures that can prevent a system from booting, the same basic procedure can be taken to bring the system up. When a system fails to boot, you should first try to identify the failure by the evidence left behind on the screen and then attempt to repair the problem (for example, by turning on a drive that was accidentally powered off). If the problem is one that cannot be repaired (such as data errors on the boot disk), boot the system from an alternate boot disk (containing a mirror of the root volume) so that the damage can be repaired or the failing disk can be replaced.

This section outlines some possible failures and provides instructions on the corrective actions.

B.4.1 Failures in UNIX Partitioning

Once the boot program has loaded, it attempts to access the boot disk through the normal UNIX partition information. If this information is damaged, the boot program fails with an error:

```
File just loaded does not appear to be executable
```

If such a message is displayed during the boot attempt, you should boot the system from an alternate boot disk. While booting, most disk drivers display errors on the console about the invalid UNIX partition information on the failing disk. The messages looks similar to:

```
WARNING: unable to read label
WARNING: corrupt label_sdo
```

This indicates that the failure was due to an invalid disk partition. You can attempt to re-add the disk as described in Section B.6.1, “Re-Adding a Failed Boot Disk.” However, if the reattach fails, then the disk will need to be replaced as described in Section B.6.2, “Replacing a Failed Boot Disk.”

B.4.2 Failures Accessing the Boot Device

Early in the boot process, immediately following system initialization, the messages may look similar to:

```
SCSI device 0,0 is not responding

Can’t open boot device
```

This means that the system PROM was unable to read the boot program from the boot drive. Common causes for this problem are:

- The boot disk is not powered on.
- The SCSI bus is not terminated.
- 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, and 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, remove the disk from the bus and replace it.

If no hardware problems are found, the error is probably due to data errors on the boot disk. To repair this problem, attempt to boot the system from an alternate boot disk (containing a mirror of the root volume). If you are unable to boot from an alternate boot disk, there is still some type of hardware problem. Similarly, if switching the failed boot disk with an alternate boot disk fails to enable the system to boot, this also indicates hardware problems.

B.4.3 Failures Due to Incorrect Entries in /etc/vfstab

When the root disk is encapsulated and put under VxVM control, as part of the normal encapsulation process, volumes are created for all of the partitions on the disk. VxVM modifies the /etc/vfstab file to use the corresponding volumes instead of the disk partitions. Take care while editing the /etc/vfstab file manually. The most important entries are those corresponding to / and /usr. The vfstab file that existed prior to Volume Manager installation is saved in /etc/vfstab.prevm.

B.4.3.1 Damaged / Entry in /etc/vfstab

If the entry in the /etc/vfstab for / file is lost or is incorrect, the system boots in single-user mode. Messages similar to the following appear on the console:

```
File just loaded does not appear to be executable
```

You should run fsck:

```
fsck /dev/vx/rdsk/rootvol
```

At this point in the boot process, / is not yet mounted read/write. Since the entry in the /etc/vfstab file was either incorrect or deleted, mount / as read/write manually, as shown in the following command:

```
mount -o remount /dev/vx/dsk/rootvol
```

After mounting / as read/write, exit the shell. The system will ask for the run level. For multi-user mode, enter run level 3:

```
ENTER RUN LEVEL (0-6,s or S): 3
```

Restore the entry in the /etc/vfstab file for / after the system boots.

B.4.3.2 Damaged /usr Entry in /etc/vfstab

The /etc/vfstab file will have an entry for /usr only if /usr is mounted on a disk partition. After encapsulation of the disk containing the /usr partition, VxVM will change the entry in /etc/vfstab to use the corresponding volume.

If the /usr entry is lost from /etc/vfstab, the system cannot be booted (even if you have mirrors of the /usr volume).

In such cases, boot the system from the CD-ROM and restore the /etc/vfstab file. (Refer to the steps in Section B.3.1, "Repairing Root (/) or /usr File Systems on Volumes.")

B.4.4 Failures Due to Missing or Damaged /etc/system

Note – Do not edit any entries in the /etc/system file that are added by VxVM. All VxVM entries are enclosed with *vxvm_START and *vxvm_END.

Make a copy of the /etc/system file in the root file system before making any changes to it. The saved system file can then be specified to the boot program if changes to the new /etc/system file are incorrect. To specify the saved system file to the boot program, boot the system with the command boot -a. When the system asks for the name of the system file, enter the path of the saved system file.

B.4.4.1 /etc/system Copy Not Available

If the /etc/system file is damaged and the saved copy of the system file is not available, the system cannot be booted with the VxVM rootability feature on. The system can be booted without VxVM rootability (that is, without the rootvol being the /) if /usr is not a volume.

To boot the system without VxVM rootability, follow the steps outlined in Section B.3.1, "Repairing Root (/) or /usr File Systems on Volumes" to:

1. **Start the Volume Manager from the CD-ROM.**
2. **Run the** fixmountroot **command.**
3. **Make and mount** /tmp/rootvol**.**

   You can now edit the file /tmp/rootvol/etc/system and perform any other necessary repairs.

4. **After booting the system, make the following entries in the `/etc/system` file:**

```
* vxvm_START
rootdev:/pseudo/vxio@0:0
set vxio:vol_rootdev_is_volume=1
* vxvm_END
```

You should also forceload all of the drivers required for the root mirror disks. To do this, edit the `/etc/system` file so that it contains a line of the following form for each driver:

```
forceload: drv/driver_name
```

You can obtain the driver names for these disks by doing a long listing on `/dev/dsk/`*root_device*. An example of a driver name would be `io-unit`.

B.4.4.2 `/etc/system` Copy Available

If the `/etc/system` file is damaged and a saved copy of the `etc/system` file is available, the system can be booted with VxVM rootability.

To boot the system with VxVM rootability, boot the system with the following command and responses (pressing Return to accept defaults for all prompts *except* the root device name):

```
ok boot -a
 .
 .
Rebooting with command: -a
Boot device: /iommu/sbus/espdma/esp/sd@5,0   File and args: -a
Enter filename [/kernel/unix]:
Name of system file [/etc/system.sav]:
Name of default directory for modules [/kernel /usr/kernel]:
Enter name of device instance number file [/etc/path_to_inst]:
root file system type [ufs]:
Enter physical name of root device
[/iommu.....................................]:/pseudo/vxio@0:0
```

B.4.4.3 `/etc/system` Not Available and `/usr` is Volume

If the `etc/system` file is damaged or lost and a backup copy is not available, and `/usr` is a volume, you must boot the system from the CD-ROM (using the steps outlined in Section B.3.1, “Repairing Root (/) or /usr File Systems on Volumes”). Once this is done, mount the root volume and edit the `etc/system` file on it. Create the following entries in the `etc/system` file:

```
* vxvm_START
rootdev:/pseudo/vxio@0:0
set vxio:vol_rootdev_is_volume=1
set vxio:vol_swapdev_is_volume=1
* vxvm_END
```

You should also forceload all of the drivers required for the root mirror disks (as described previously). After these changes, reboot the system from the same root partition on which the system file was restored.

B.4.5 Failures Due to Booting From Unusable or Stale Plexes

If a disk is unavailable when the system is running, any mirrors of volumes that reside on that disk will become stale, meaning that the data on that disk is out of date relative to the other mirrors of that volume. During the boot process, the system accesses only one copy of the `root` volume (the copy on the boot disk) until a complete configuration for this volume can be obtained. If it turns out that the plex of this volume that was used for booting is stale, the system must be rebooted from an alternate boot disk that contains non-stale plexes. This problem can occur, for example, if the system was booted from one of the disks made bootable by VxVM with the original boot disk turned off. The system will boot normally, but the plexes that reside on the unpowered disk will be stale. If the system reboots from the original boot disk with the disk turned back on, the system will boot using that stale plex.

Another possible problem can occur if errors in the VxVM headers on the boot disk prevent VxVM from properly identifying the disk. In this case, VxVM will not know the name of that disk. This is a problem because plexes are associated with disk names, so any plexes on the unidentified disk are unusable.

A problem can also arise if the root disk experiences a failure that affects the root volume’s plex. At the next boot attempt, the system will still expect to be able to use the failed root plex for booting. If the root disk was mirrored at the time of the failure, you can specify an alternate root disk (with a valid root plex) for booting.

If any of these situations occurs, the VxVM utility `vxconfigd` will notice it when it is configuring the system as part of the `init` processing of the boot sequence. `vxconfigd` will display a message describing the error and what can be done about it, then it will halt the system. For example, if the plex `rootvol-01` of the root volume `rootvol` on disk `rootdisk` is stale, `vxconfigd` might 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: c0t1d0s2
vxvm:vxconfigd:    Error: System startup failed
The system is down.
```

This informs the administrator that the alternate boot disk named `disk01` contains a usable copy of the root plex and should be used for booting. When this message is displayed, you should reboot the system from the alternate boot disk.

Once the system has booted, the exact problem needs to be determined. If the plexes 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 or the disk failed, you will need to re-add or replace the disk.

If the plexes on the boot disk are unavailable, you should receive mail from VxVM utilities describing the problem. Another way to determine 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 `rootdisk` (such as due to media failures or accidentally overwriting the VxVM private region on the disk), `vxdisk list` would show output such as:

```
 DEVICE     TYPE       DISK       GROUP    STATUS

   -          -        rootdisk   rootdg   failed was: c0t3d0s2
 c0t1d0s2    sliced    disk01     rootdg   ONLINE
```

B.5 Hot-Relocation and Boot Disk Failures

If the boot (root) disk is mirrored and it experiences a failure, hot-relocation automatically attempts to replace the failed root disk mirror with a new mirror. To do this, hot-relocation uses a surviving mirror of the root disk to create a new mirror on either a spare disk or a disk with sufficient free space. This ensures that there are always at least two mirrors of the root disk that can be used for booting. The hot-relocation daemon also calls the `vxbootsetup` utility, which configures the disk with the new mirror as a bootable disk.

Hot-relocation may fail for a root disk if the `rootdg` disk group does not contain sufficient spare or free space to accommodate the volumes from the failed root disk. The `rootvol` and `swapvol` volumes require contiguous disk space. If the root volume and other volumes on the failed root disk cannot be relocated to the same new disk, each of these volumes can be relocated to a different disk. Mirrors of `rootvol` and `swapvol` volumes must be cylinder-aligned, so they can only be created on disks with enough space to allow their subdisks to begin and end on cylinder boundaries; hot-relocation will fail if such disks are not available.

B.6 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 (using `vxdiskadm`). Data that is not critical for booting the system is accessed by the Volume Manager only after the system is fully operational, so it doesn't matter where that data is located—the Volume Manager can find it. However, 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 problem:

- If the error(s) are transient or correctable, 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 are sufficient to make a disk re-usable and thus a candidate for re-addition.
- If the disk has truly failed, you must replace it.

The following sections describe how to re-add or replace a failed boot disk.

B.6.1 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, consider a system that has two disks, `disk01` and `disk02`, which are normally mapped into the system configuration during boot as disks `c0t0d0s2` and `c0t1d0s2`, respectively. A failure has caused `disk01` to become detached. You can confirm this by listing the disks with the `vxdisk` utility:

```
vxdisk list
```

This would result in the following output:

```
DEVICE       TYPE      DISK      GROUP      STATUS
c0t0d0s2     sliced    -         -          error
c0t1d0s2     sliced    disk02    rootdg     online
-            -         disk01    rootdg     failed was:c0t0d0s2
```

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. The device `c0t0d0s2` may not be listed at all; this would occur if the disk failed totally and the disk controller did not notice it on the bus.

In some cases, the `vxdisk list` output may differ. 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), but no errors in the VxVM private area, the output of the `vxdisk list` command resembles the following:

```
DEVICE      TYPE      DISK      GROUP      STATUS
c0t0d0s2    sliced    disk01    rootdg     online
c0t1d0s2    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, you must detach the failing disk from its device manually. This is done using the “Remove a disk for replacement” function of the `vxdiskadm` utility (see the `vxdiskadm(1M)` manual page or the *Sun StorEdge Volume Manager 2.6 User’s Guide* for more information about `vxdiskadm`). Once the disk is detached from the device, you can follow any special procedures for correcting the problem (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 previous examples, this would mean replacing `disk01` with the device `c0t0d0s2`.

If hot-relocation is enabled when a mirrored boot disk fails, it attempts to create a new mirror and remove the failed subdisks from the failing boot disk. If a re-add succeeds after a successful hot-relocation, the root and/or other volumes affected by the disk failure no longer exist on the re-added disk. However, the re-added disk can still be used for other purposes.

If a re-add of the disk fails, replace the disk.

B.6.2 Replacing a Failed Boot Disk

When a boot disk needs to be replaced, you should first boot the system off an alternate boot disk. If the failing disk is not detached from its device, manually detach it using the “Remove a disk for replacement” function of `vxdiskadm` (see the `vxdiskadm(1M)` manual page or the *Sun StorEdge Volume Manager 2.6 User’s Guide* for more information about `vxdiskadm`). Once the disk is detached, shutdown the system and replace the hardware.

The replacement disk should have at least as much storage capacity as was in use on the disk being replaced. The replacement disk should be large enough so that the region of the disk for storing subdisks can accommodate all subdisks of the original disk at their current disk offsets. To determine the minimum size of a replacement disk, you need to determine how much space was in use on the disk that failed.

To approximate the size of the replacement disk, type:

```
vxprint -st -e 'sd_disk="diskname"'
```

From the resulting output, 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. When the hardware replacement is complete, boot the system and use `vxdiskadm`’s “Replace a failed or removed disk” function to replace the failing disk with the new device that was just added.

B.7 Reattaching Disks

A disk reattach operation may be appropriate if a disk has experienced a full failure and hot-relocation is not possible, or if the Volume Manager is started with some disk drivers unloaded and unloadable (causing disks to enter the failed state). If the problem is fixed, you may be able to use the `vxreattach` command to reattach the disks without plexes being flagged as stale, as long as the reattach happens before any volumes on the disk are started.

The `vxreattach` command is called as part of disk recovery from the `vxdiskadm` menus and during the boot process. If possible, `vxreattach` will reattach the failed disk media record to the disk with the same device name in the disk group in which it was located before and will retain its original disk media name. After a reattach takes place, recovery may or may not be necessary. The reattach may fail if the original (or another) cause for the disk failure still exists.

The `vxreattach -c` command checks whether a reattach is possible, but does not actually perform the operation. Instead, it displays the disk group and disk media name where the disk can be reattached.

Refer to the `vxreattach(1M)` manual page for more information on the `vxreattach` command.

B.8 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.8.1 General Reinstallation 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 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. You can replace any other disks to be replaced by following the procedures in the *Sun StorEdge Volume Manager 2.6 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. By having the root disk under VxVM control and creating mirrors of the root disk contents, you can eliminate 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.

B.8.2 Reinstallation and Reconfiguration Procedures

To reinstall the system and recover the VxVM configuration, perform the following procedure (these steps are described in detail in the sections that follow):

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. **Clean up the VxVM configuration.**

   This includes restoring any information in volumes affected by the failure or reinstallation, and recreating system volumes (`rootvol`, `swapvol`, `usr`, etc.).

B.8.2.1 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.

Note – Several of the *automatic* options for installation access disks other than the root disk without requiring confirmation from the administrator. Therefore, you should disconnect all other disks (containing volumes) from the system prior to reinstalling the operating system.

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 disk, it may still be recoverable. Removing the second disk ensures that the home file system remains intact.

B.8.2.2 Reinstalling the Operating System

Once any failed or failing disks have been replaced and disks not involved with the reinstallation have been detached, reinstall the operating system (as described in your operating system documentation). Install the 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 – During reinstallation, you may have the opportunity to change the host ID. You should keep the existing host ID, as the sections that follow assume that you have not changed your host ID.

B.8.2.3 Reinstalling the Volume Manager

Installation of the Volume Manager involves:

1. Loading VxVM from CD-ROM
2. Initializing the Volume Manager

To reinstall the Volume Manager, follow the instructions for loading the Volume Manager (from CD-ROM) in the *Sun StorEdge Volume Manager 2.6 Installation Guide*.

Note – If you want to reconstruct the Volume Manager configuration left on the non-root disks, *do not* initialize the Volume Manager (using `vxinstall`) after the reinstallation.

Use `vxserial` to install the VxVM license key (see the `vxserial (1M)` manual page).

B.8.2.4 Recovering the Volume Manager Configuration

Once the VxVM package has been loaded, recover the Volume Manager configuration as follows:

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 typing the following command:**

```
shutdown -g0 -iS -y
```

5. **When prompted to do so, enter the password and press Return to continue.**
6. **Remove the files involved with installation that were created when you loaded VxVM but are no longer needed. To do this, type:**

```
rm -rf /etc/vx/reconfig.d/state.d/install-db
```

7. **Once these files are removed, you should start some VxVM I/O daemons. Start the daemons by typing:**

```
vxiod set 10
```

8. **Start the Volume Manager configuration daemon, `vxconfigd`, in disabled mode by typing:**

```
vxconfigd -m disable
```

9. **Initialize the `vxconfigd` daemon by typing:**

```
vxdctl init
```

10. **Enable `vxconfigd` by typing:**

```
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 disks 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 mirrors are to be replaced, follow the replacement procedures in the *Sun StorEdge Volume Manager 2.6 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 volumes or mirrors on that disk (or other disks no longer attached to the system) are now inaccessible. If a volume had only one plex (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 `/usr` file system are *not* located on volumes any longer. To correct these problems, follow the instructions in Section B.8.2.5, "Configuration Cleanup."

B.8.2.5 Configuration Cleanup

The following sections describe how to clean up the configuration of your system after reinstallation of the Volume Manager.

The following types of cleanup are described:

- Rootability cleanup
- Volume cleanup
- Disk cleanup

These sections are followed by reconfiguration information:

- Rootability reconfiguration
- Final reconfiguration

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 (and any other disk involved in the system boot process) was under Volume Manager control. The volumes to remove are:

- `rootvol`, which contains the `root` file system
- `swapvol`, which contains the `swap` area
- `usr`, which contains the `/usr` file system

To remove the root volume, use the `vxedit` command, as follows:

```
vxedit -fr rm rootvol
```

Repeat this command, using `swapvol` and `usr` in place of `rootvol`, to remove the `swap` and `usr` volumes.

Volume Cleanup

After completing the rootability cleanup, you must determine which volumes need to be restored from backup. The volumes to be restored include those with all mirrors (all copies of the volume) residing 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 of a volume exist on reinitialized or removed disks, these mirrors must be removed. The mirrors can be re-added later.

To restore the volumes:

1. **Establish which VM disks have been removed or reinstalled by typing:**

```
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:

```
DEVICETYPEDISKGROUPSTATUS
c0t0d0s2sliced--error
c0t1d0s2
    sliceddisk02rootdgonline
c0t2d0s2
    sliceddisk03rootdgonline
-   -disk01rootdgfailed was: c0t0d0s2
```

The display shows that the reinstalled root device, `c0t0d0s2,` is not associated with 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 (`c0t1d0s2` and `c0t2d0s2`). The former `disk01`, which was the VM disk associated with the replaced disk device, is no longer associated with the device (`c0t0d0s2`).

If other disks (with volumes or mirrors on them) had been removed or replaced during reinstallation, those disks would also have a disk device listed 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, locate all the mirrors on failed disks by typing:**

```
vxprint -sF “%vname” -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, type:**

```
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 `v01` with only one plex resides on the reinstalled disk named `disk01`. The `vxprint -th v01` command produces the following display:

```
V   NAME         USETYPE   KSTATE    STATE     LENGTH   READPOL    PREFPLEX
PL  NAME         VOLUME    KSTATE    STATE     LENGTH   LAYOUT     NCOL/WID   MODE
SD  NAME         PLEX      DISK      DISKOFFS  LENGTH   [COL/]OFF  DEVICE     MODE

v   v01          fsgen     DISABLED  ACTIVE    24000    SELECT      -
pl  v01-01       v01       DISABLED  NODEVICE  24000    CONCAT      -          RW
sd  disk01-06    v01-01    disk01    245759    24000    0          c1t5d1     ENA
```

The only plex of the volume is shown in the line beginning with `pl`. The `STATE` field for the plex named `v01-01` is `NODEVICE`. The plex has space on a disk that has been replaced, removed, or reinstalled. Therefore, the plex is no longer valid and must be removed.

Since `v01-01` was the only plex 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, as you will need it for the backup procedure.

4. **To remove the volume `v01`, use the `vxedit` command:**

```
vxedit -r rm v01
```

It is possible that only part of a plex is located on the failed disk. If the volume has a striped plex associated with it, the volume is divided among several disks. For example, the volume named `v02` has one striped plex striped across three disks, one of which is the reinstalled disk `disk01`. The output of the `vxprint -th v02` command returns:

```
V   NAME        USETYPE   KSTATE    STATE     LENGTH   READPOL    PREFPLEX
PL  NAME        VOLUME    KSTATE    STATE     LENGTH   LAYOUT     NCOL/WID   MODE
SD  NAME        PLEX      DISK      DISKOFFS  LENGTH   [COL/]OFF  DEVICE     MODE

v   v02         fsgen     DISABLED  ACTIVE    30720     SELECT    v02-01
pl  v02-01      v02       DISABLED  NODEVICE  30720     STRIPE    3/128      RW
sd  disk02-02   v02-01    disk02    424144    10240     0/0       c1t2d0     ENA
sd  disk01-05   v02-01    disk01    620544    10240     1/0       c1t2d1     DIS
sd  disk03-01   v02-01    disk03    620544    10240     2/0       c1t2d2     ENA
```

The display shows three disks, across which the plex `v02-01` 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 plex named `v02-01` has a state of `NODEVICE`. Since this is the only plex of the volume, the volume is invalid and must be removed. If a copy of `v02` exists on the backup media, it can be restored later. Keep a record of the volume name and length of any volume you intend to restore from backup.

5. **Use the `vxedit` command to remove the volume, as described in Step 4.**

   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 (`disk01`) and another plex on a valid disk (`disk02`) would look like this:

```
V   NAME        USETYPE   KSTATE    STATE     LENGTH   READPOL     PREFPLEX
PL  NAME        VOLUME    KSTATE    STATE     LENGTH   LAYOUT      NCOL/WID  MODE
SD  NAME        PLEX      DISK      DISKOFFS  LENGTH   [COL/]OFF   DEVICE    MODE

v   v03         fsgen     DISABLED  ACTIVE    30720     SELECT     -
pl  v03-01      v03       DISABLED  ACTIVE    30720     CONCAT     -         RW
sd  disk02-01   v03-01    disk01    620544    30720     0          c1t3d0    ENA
pl  v03-02      v03       DISABLED  NODEVICE  30720     CONCAT     -         RW
sd  disk01-04   v03-02    disk03    262144    30720     0          c1t2d2    DIS
```

This volume has two plexes, `v03-01` and `v03-02`. The first plex (`v03-01`) does not use any space on the invalid disk, so it can still be used. The second plex (`v03-02`) uses space on invalid disk `disk01` and has a state of `NODEVICE`. Plex `v03-02` 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 want to create another plex later.

6. **To remove an invalid plex, you must dissociate and remove the plex from the volume using the** `vxplex` **command. To remove the plex** `v03-02`**, type:**

```
vxplex -o rm dis v03-02
```

7. **Once all the volumes have been cleaned up, clean up the disk configuration as described in the following section "Disk Cleanup."**

Disk Cleanup

Once all invalid volumes and 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 disk `disk01`, type:

```
vxdg rmdisk disk01
```

If the `vxdg` command returns an error message, some invalid mirrors exist. Repeat the processes described in the section "Volume Cleanup" until all invalid volumes and mirrors are removed.

Rootability Reconfiguration

Once you have removed all the invalid disks, you can add the replacement or reinstalled disks to Volume Manager control. If the root disk was originally under VxVM control or you now want 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`):

```
vxdiskadm
```

From the `vxdiskadm` main menu, select menu item `2` (`Encapsulate a disk`). Follow the instructions and encapsulate the root disk for the system. For more information, see the *Sun StorEdge Volume Manager 2.6 User's Guide*.

When the encapsulation is complete, reboot the system to multi-user mode.

Final Reconfiguration

Once the root disk is encapsulated, add any other disks that were replaced using `vxdiskadm`. If the disks were reinstalled during the operating system reinstallation, they should be encapsulated; otherwise, they can simply be added.

Once all the disks have been added to the system, you can re-create any volumes that were completely removed as part of the configuration cleanup and restore their contents from backup using normal backup/restore procedures. You can re-create the volume using `vxassist` or the Visual Administrator interface.

To re-create the volumes `v01` and `v02` using the `vxassist` command, type:

```
vxassist make v01 24000
vxassist make v02 30720 layout=stripe nstripe=3
```

You can re-create these mirrors for any volumes that had plexes removed as part of the volume cleanup by following the instructions for mirroring a volume (via `vxassist` or the Visual Administrator), as described in the *Sun StorEdge Volume Manager 2.6 User's Guide*.

To replace the plex removed from volume `v03` using `vxassist`, type:

```
vxassist mirror v03
```

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.

B.9 Plex and Volume States

The following sections describe plex and volume states.

B.9.1 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 `vxmake plex` command; a 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

Plexes that are associated with a volume have one of the following states:

- EMPTY
- CLEAN
- ACTIVE
- STALE
- OFFLINE
- TEMP
- TEMPRM
- TEMPRMSD
- IOFAIL

A Dirty Region Logging or RAID-5 log plex is a special case, as its state is always set to `LOG`.

B.9.1.1 EMPTY Plex State

Volume creation sets all plexes associated with the volume to the `EMPTY` state to indicate that the plex is not yet initialized.

B.9.1.2 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 an 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.9.1.3 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 system running well, ACTIVE should be the most common state you see for any volume's plexes.

B.9.1.4 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 an operation sets the state of the plex to STALE.

A `vxplex att` operation recovers the contents of a STALE plex 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 a `vxplex det` operation.

B.9.1.5 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 `vxvol start` operation.

B.9.1.6 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 `vxvol start` will dissociate plexes in the TEMP state.

B.9.1.7 TEMPRM Plex State

A TEMPRM plex state resembles a TEMP state except that at the completion of the operation, the 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.9.1.8 TEMPRMSD Plex State

The TEMPRMSD plex state is used by `vxassist` when attaching new plexes. If the operation does not complete, the plex and its subdisks are removed.

B.9.1.9 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.9.2 The Plex State Cycle

The changing of plex states accompanies normal operations. Deviations in plex state indicate abnormalities that VxVM must normalize. At system startup, volumes are automatically started and the `vxvol 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 vxvol start makes them ACTIVE) indicates a normal shutdown and optimizes startup.

B.9.3 Plex Kernel State

The *plex kernel state* indicates the accessibility of the plex. The plex kernel state is monitored in the volume driver and enables a plex to have an offline (DISABLED), maintenance (DETACHED), or online (ENABLED) mode of operation.

The following are plex kernel states:

- 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. Plex operations and ioctl functions are accepted.
- ENABLED — A write request to the volume will be reflected to the plex. A read request from the volume will be satisfied from the plex.

Note – No user intervention is required to set these states; they are maintained internally. On a system that is operating properly, all plexes are enabled.

B.9.4 Volume States

There are several volume states, some of which are similar to plex states:

- CLEAN — The volume is not started (kernel state is DISABLED) and its plexes are synchronized.
- ACTIVE — The volume has been started (kernel state is currently ENABLED) or was in use (kernel state 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, the plexes cannot be guaranteed to be consistent, but will be made consistent when the volume is started.
- EMPTY — The volume contents are not initialized. The kernel state is always DISABLED when the volume is EMPTY.
- SYNC — The volume is either in read-writeback recovery mode (kernel state is currently ENABLED) or was in read-writeback mode when the machine was rebooted (kernel state is DISABLED). With read-writeback recovery, plex consistency is recovered by reading data from blocks of one plex and writing the data to all other writable plexes. If the volume is ENABLED, the plexes are being

resynchronized via the read-writeback recovery. If the volume is DISABLED, the plexes were being resynchronized via read-writeback when the machine rebooted and therefore still need to be synchronized.

- NEEDSYNC — The volume will require a resynchronization operation the next time it is started.

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, 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.9.4.1 RAID-5 Volume States

RAID-5 volumes have their own set of volume states:

- CLEAN — The volume is not started (kernel state is DISABLED) and its parity is good. The RAID-5 plex stripes are consistent.
- ACTIVE — The volume has been started (kernel state is currently ENABLED) or was in use (kernel state was ENABLED) when the machine was rebooted. If the volume is currently ENABLED, the state of its RAID-5 plex at any moment is not certain (since the volume is in use). If the volume is currently DISABLED, the parity cannot be guaranteed to be synchronized.
- EMPTY — The volume contents are not initialized. The kernel state is always DISABLED when the volume is EMPTY.
- SYNC — The volume is either undergoing a parity resynchronization (kernel state is currently ENABLED) or was having its parity resynchronized when the machine was rebooted (kernel state is DISABLED).
- NEEDSYNC — The volume will require a parity resynchronization operation the next time it is started.
- REPLAY — The volume is in a transient state as part of a log replay. A log replay occurs when it becomes necessary to use logged parity and data.

B.9.5 Volume Kernel State

The *volume kernel state* indicates the accessibility of the volume. The following are volume kernel states:

- 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.

Index

A

adding disks, 3-4
  format, 3-4
associating log subdisks, 4-14
  vxsd, 4-15
associating mirrors
  vxmake, 4-20
associating plexes
  vxmake, 4-20
associating subdisks
  vxmake, 4-13
  vxsd, 4-13
autoboot flag, B-2

B

backups, 4-57, B-2
  mirrors, 4-19
  vxassist, 4-57
boot disk
  failure, B-16
  failure and hot-relocation, B-16
  re-adding, B-16, B-17
  replacing, B-16, B-18
boot process, B-2
booting
  after failure, B-3

C

changing volume attributes, 4-56
checkpoint, 4-49
columns, in striping, 1-11
command-line utilities, 3-3
concatenation, 1-5
configuration guidelines, 2-1
copying mirrors
  vxplex, 4-27
creating disk groups, 3-18
  vxdg, 3-18
creating mirrors
  vxmake, 4-18
creating RAID-5 volumes, 4-45
creating subdisks, 4-11
  vxmake, 4-11
creating volumes, 4-44, 4-45
  manually, 4-2
  vxassist, 4-1

D

daemons, 1-33
  configuration, 4-6, A-3
  hot-relocation, 1-26
  Volume Manager, 1-33
  vxrelocd, 3-11
data
  preserving, B-2
  redundancy, 1-15
data assignment, 2-1
defaults file
  vxassist, 4-4
degraded mode, 4-47
deporting
  disk groups, 3-19

description file, 4-7
device name, 3-1
Dirty Region Logging, 1-29
  guidelines, 2-5
  log subdisks, 4-14
disk arrays, 1-10, 1-36
disk failures, 4-47
  protecting data, B-1
disk group utilities, 3-3
disk groups, 1-4, 3-2, 3-18
  creating, 3-18
  deporting, 3-19, 3-21
  importing, 3-19, 3-21
  moving, 3-19, 3-21
  moving between systems, 3-19
  removing, 3-19
  renaming, 3-22
  using, 3-18
disk media name, 1-3, 3-1
disk names, 3-1, 3-3
disk utilities, 3-3
disks
  adding, 3-4
  boot disk, B-16
  detached, 3-13
  encapsulation, 1-31, 3-25, B-1, B-4
  failure, 3-9, 4-47
    and hot-relocation, 1-26, 3-9
    protecting data, B-1
  hot-relocation spares, 3-12
  initialization, 3-4
  moving, 3-9
  physical, 1-2
  re-adding, B-16
  reattaching, B-19
  removing, 3-7
  replacing, 3-15, 3-16, B-16
  root disk, 1-31, B-1, B-3, B-4, B-16
  VM disks, 1-3
  volatile, 3-26
displaying subdisks
  vxprint, 4-12
dissociating mirrors
  vxplex, 4-21

E

encapsulation, 1-31, 3-25, B-1, B-4

F

failed disks, 3-9
  detecting, 3-12
failures, 4-50, B-9
  and recovery procedures, B-9
  disk, 4-47
  system, 4-46
forcibly starting volumes, 4-54
format utility, 3-4

G

getting performance data, 2-10
guidelines
  Dirty Region Logging, 2-5
  hot-relocation, 2-8
  mirroring, 2-5
  mirroring and striping, 2-7
  RAID-5, 2-7
  striping, 2-3

H

hot-relocation, 1-26, 3-9
  boot disk, B-16
  designating spares, 1-27
  guidelines, 2-8
  modifying vxrelocd, 3-11

I

I/O
  statistics, 2-12
    obtaining, 2-10
  tracing, 2-11, 2-15
I/O daemon, 1-34
importing
  disk groups, 3-19
information, 4-16
initializing disks, 3-4
interfaces
  Volume Manager, 1-35

J

joining subdisks
    vxsd, 4-17

L

layout
    left-symmetric, 1-18
listing mirrors
    vxprint, 4-22
log plexes, 4-41
log subdisks, 1-29, 2-6, 4-14
    associating, 4-14
logging, 1-20
logs, 4-42, 4-51

M

manipulating subdisks, 4-51
mirror attributes
    changing, 4-23
mirroring, 1-15, 1-37, 2-3, 2-6
    guidelines, 2-5
mirrors, 1-5, 1-8
    backup using, 4-19
    creating, 4-18
    displaying, 4-22
    dissociating, 4-21
    offline, 4-24, 4-39
    online, 4-39
    recover, 3-14
    removing, 4-21
moving disk groups
    vxdg, 3-19
    vxrecover, 3-20
moving disks, 3-9
moving mirrors, 4-26
    vxplex, 4-26
moving subdisks
    vxsd, 4-16

N

name
    disk access, 1-2
    disk media, 1-3
nopriv, 3-25
    devices, 3-26

O

objects
    physical, 1-2
    Volume Manager, 1-3, 1-9
OFFLINE, 4-24
online backup, 4-57
ownership, 4-34

P

parity, 1-16, 4-49
parity recovery, 4-49
partitions, 1-2
performance, 2-7
    guidelines, 2-1
    management, 2-1
    monitoring, 2-9
    optimizing, 2-1
    priorities, 2-9
performance data, 2-10
    getting, 2-10
    using, 2-11
permission, 4-35
physical disks, 1-2
physical objects, 1-2
plex kernel states, B-33
    DETACHED, B-33
    DISABLED, B-33
    ENABLED, B-33
plex states, B-30
    ACTIVE, B-31
    CLEAN, B-31
    EMPTY, B-30
    IOFAIL, B-32
    OFFLINE, B-31
    STALE, B-31
    TEMP, B-32
    TEMPRM, B-32
plex states cycle, B-32
plexes, 1-5, 4-42
    and volume, 1-7
    as mirrors, 1-8
    attach, 4-8

attaching, 4-24, 4-25
changing information, 4-22
copying, 4-27
creating, 4-18
definition, 1-5
detach, 4-8
detaching, 4-24
displaying, 4-22
listing, 4-22
moving, 4-26
offline, 4-39
online, 4-39
striped, 1-11
private region, 3-2
public region, 3-2
putil, 4-22

R

RAID, 1-10, 1-37
RAID-0, 1-11, 1-37
RAID-1, 1-15, 1-37
RAID-2, 1-37
RAID-3, 1-38
RAID-4, 1-40
RAID-5, 1-40, 2-7, 4-41, 4-42, 4-44, 4-45, 4-46, 4-47, 4-49, 4-51, 4-53, 4-54, 4-56
guidelines, 2-7
recovery, 4-48, 4-55
RAID-5 plexes, 4-41
RAID-5 volumes, 4-55
read
policies, 2-4
re-adding disks, B-16
reattaching disks, B-19
reconfiguration procedures, B-20
recontructing-read, 4-47
recovery, 4-55
logs, 4-50
procedures, B-9
RAID-5 volumes, 4-48, 4-55
volumes, 4-40
reinstallation, B-20, B-22
removing disk groups, 3-19
vxdg, 3-19
removing disks, 3-7
vxdg, 3-7
removing mirrors, 4-21
removing subdisks
vxedit, 4-12
renaming disk groups, 3-22
replacing disks, 3-15, B-16
vxdiskadm, 3-15
resynchronization
and Oracle databases, 1-30
volume, 1-28
root disk, 1-31, B-1, B-3, B-4, B-16
root file system, B-4
root volume restrictions, 1-32
root volumes
booting with, 1-32
rootability, 1-31
cleanup, B-24
rootdg, 1-4, 3-2
renaming, 3-22

S

SNAPDONE, 4-57
snapshot, 4-57, 4-58
snapstart, 4-57
snapwait, 4-57
spanning, 1-5
special devices
using, 3-24
special encapsulations
vxdisk, 3-25
spindles, synchronized, 1-39
splitting subdisks
vxsd, 4-17
standard disk devices, 3-1
starting volumes, 4-53
states
plex, B-30
volume, B-33
stripe column, 1-11
stripe units, 1-11
striped plex, 1-11
striping, 1-11, 1-37, 2-2, 2-6
guidelines, 2-3
subdisks, 1-4, 4-1
associating, 4-12, 4-13
changing information, 4-16
creating, 4-11
definition, 1-4
displaying, 4-12

dissociating, 4-15
joining, 4-17
log, 1-29, 4-14
moving, 4-16
operations, 4-11
removing, 4-12
splitting, 4-17
swap volume restrictions, 1-32
system failures, 4-46

T

tracing I/O, 2-11
vxtrace, 2-11
tunables, 2-15
tuning
Volume Manager, 2-15
tutil, 4-22

U

unusable volumes, 4-53
using disk groups
vxassist, 3-18
using disks, 3-4
using I/O statistics, 2-12
using performance data, 2-11
using special devices, 3-24
utility descriptions, 4-2
vxassist, 4-2
vxdctl, 4-6
vxedit, 4-6
vxmake, 4-7
vxmend, 4-8
vxplex, 4-8
vxprint, 4-9
vxsd, 4-9
vxstat, 4-9
vxvol, 4-10

V

Visual Administrator, 1-35, 3-3
VM disks, 1-3
definition, 1-3
volume kernel states, B-34
DETACHED, B-34
DISABLED, B-34
ENABLED, B-34
Volume Manager, 1-1
and RAID, 1-10
daemons, 1-33
description, 1-1
interfaces, 1-35
objects, 1-3, 1-9
overview, 1-1
rootability, 1-31
Volume Manager Support Operations, 3-2
Volume Manager Visual Administrator, 1-35, 3-3
volume restrictions
boot-time, 1-32
volume resynchronization, 1-28
volume states, B-33
ACTIVE, B-33, B-34
CLEAN, B-33, B-34
EMPTY, B-33, B-34
SYNC, B-33, B-34
volumes, 1-1, 1-7, 4-27, 4-34
and plexes, 1-7
changing information, 4-34
cleanup, B-24
creating, 4-28
definition, 1-1, 1-7
displaying, 4-33
error policies, 4-34
estimating size, 4-31
initializing, 4-30
kernel state, 4-10
layout, 4-41
mirroring, 4-40
operations, 4-10
permissions, 4-34
read policy, 4-37
recovery, 4-40
removing, 4-32
resizing, 4-35
starting, 4-38
stopping, 4-38
vxassist, 3-16, 3-18, 4-2, 4-3, 4-28, 4-35, 4-40, 4-57, 4-58, B-2
backup, 4-57
creating volumes, 4-1
defaults, 4-4
description of, 4-2
growby, 4-36

growto, 4-36
shrinkby, 4-36
shrinkto, 4-36
snapshot, 4-57
snapstart, 4-57
snapwait, 4-57
vxconfigd, 1-32, 1-33, 4-6, A-3
vxdctl, 4-6
description of, 4-6
vxdg, 3-4, 3-7, 3-18, 3-19, 3-21
moving disk groups, 3-19
removing disk groups, 3-19
vxdisk, 3-4, 3-25
rm, 3-8
special encapsulations, 3-25
vxdiskadd, 3-2, 3-3, 3-5, 3-18
vxdiskadm, 1-31, 3-2, 3-3, 3-5, 3-8, 3-9, 3-15, 3-21
replacing disks, 3-15
vxedit, 4-6, 4-12, 4-16, 4-22, 4-23, 4-33, B-24
description of, 4-6
removing subdisks, 4-12
vxinfo, 3-15
vxiod, 1-33, 1-34
vxmake, 4-7, 4-11, 4-13, 4-18, 4-20, 4-29, 4-40, 4-44, 4-45
associating mirrors, 4-20
associating subdisks, 4-13
creating mirrors, 4-18
creating subdisks, 4-11
description of, 4-7
vxmend, 4-8, 4-24, 4-26, 4-39, 4-40
vxmirror, B-4
vxplex, 4-8, 4-19, 4-20, 4-21, 4-24, 4-25, 4-26, 4-40
copying mirrors, 4-27
description of, 4-8
dissociating mirrors, 4-21
moving mirrors, 4-26
vxprint, 3-12, 4-9, 4-12, 4-22, 4-33, 4-34
description of, 4-9
displaying subdisks, 4-12
listing mirrors, 4-22
vxreattach, B-19
vxrecover, 3-14
moving disk groups, 3-20
vxrelocd, 1-26, 3-11
modifying, 3-11
vxsd, 4-9, 4-13, 4-15, 4-16, 4-17
associating log subdisks, 4-15
associating subdisks, 4-13
description of, 4-9
joining subdisks, 4-17
moving subdisks, 4-16
splitting subdisks, 4-17
vxstat, 2-10, 2-12, 3-13, 4-9
description of, 4-9
vxtrace, 2-10, 2-11, 4-10
VxVM, 1-1
vxvol, 3-15, 4-10, 4-25, 4-30, 4-35, 4-37
description of, 4-10

W

writes
full-stripe, 1-22
parallel, 1-39
read-modify, 1-21
reconstruct, 1-24