VERITAS Volume Manager™ 3.1

Administrator’s Guide

Solaris

July 2000
30-000000-399

VERITAS

Disclaimer

The information contained in this publication is subject to change without notice. VERITAS Software Corporation makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. VERITAS Software Corporation shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Copyright

Copyright © 1999-2000 VERITAS Software Corporation. All rights reserved. VERITAS is a registered trademark of VERITAS Software Corporation in the US and other countries. The VERITAS logo and VERITAS Volume Manager are trademarks of VERITAS Software Corporation. All other trademarks or registered trademarks are the property of their respective owners.

Printed in the USA, July 2000.

VERITAS Software Corporation
1600 Plymouth St.
Mountain View, CA 94043
Phone 650–335–8000
Fax 650–335–8050
www.veritas.com

Preface

The *VERITAS Volume Manager^{TM} Administrator's Guide* provides information on how to use Volume Manager.

Audience

This guide is intended for system administrators responsible for installing, configuring, and maintaining systems under the control of the VERITAS Volume Manager.

This guide assumes that the user has a:

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

Scope

The purpose of this guide is to provide the system administrator with a thorough knowledge of the procedures and concepts involved with volume management and system administration using the Volume Manager. This guide includes guidelines on how to take advantage of various Volume Manager features, instructions on how to use Volume Manager commands to create and manipulate objects, and information on how to recover from disk failures.

Organization

This guide is organized as follows:

- Understanding Volume Manager
- Getting Started
- Volume Manager Operations
- Disk Tasks

- Volume Tasks
- Volume Manager Cluster Functionality
- Recovery
- VxVM Performance Monitoring

Using This Guide

This guide contains instructions for performing Volume Manager system administration tasks. Volume Manager administration functions can be performed through one or more of the following interfaces:

- a set of complex commands
- a single automated command (`vxassist`)
- a menu-driven interface (`vxdiskadm`)
- the Storage Administrator (graphical user interface)

This guide describes how to use the various Volume Manager command line interfaces for Volume Manager administration. Details on how to use the Storage Administrator graphical user interface can be found in the *VERITAS Volume Manager Storage Administrator Administrator's Guide.* Detailed descriptions of the Volume Manager utilities, the options for each utility, and details on how to use them are located in the Volume Manager manual pages.

Note Most of the Volume Manager commands require superuser or other appropriate privileges.

Related Documents

The following documents provide information related to the Volume Manager:

- *VERITAS Volume Manager Installation Guide*
- *VERITAS Volume Manager Release Notes*
- *VERITAS Volume Manager Hardware Notes*
- *VERITAS Volume Manager Reference Guide*

- *VERITAS Volume Manager Storage Administrator Administrator’s Guide*
- VERITAS Volume Manager manual pages

Conventions

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

Typeface	Usage	Examples
`monospace`	Computer output, files, directories, software elements such as command options, function names, and parameters	Read tunables from the `/etc/vx/tunefstab` file. See the `ls`(1) manual page for more information.
`monospace` (**`bold`**)	User input	# **`mount -F vxfs /h/filesys`**
italic	New terms, book titles, emphasis, variables replaced with a name or value	See the *User’s Guide* for details. The variable *ncsize* determines the value of...

Symbol	Usage	Examples
%	C shell prompt	
$	Bourne/Korn shell prompt	
#	Superuser prompt (all shells)	
\	Continued input on the following line; you do not type this character	# **`mount -F vxfs`** \ **`/h/filesys`**
[ ]	In a command synopsis, brackets indicates an optional argument	`ls [ -a ]`
|	In a command synopsis, a vertical bar separates mutually exclusive arguments	`mount [ suid | nosuid ]`

Contents

Understanding Volume Manager 1

Introduction

This guide is an overview of the VERITAS Volume Manager (VxVM®) that describes what Volume Manager is, how it works, how you can communicate with it through the user interfaces, and Volume Manager concepts. Related documents that provide specific information about Volume Manager are listed in the "Preface."

VERITAS Volume Manager provides easy-to-use online disk storage management for computing environments. Traditional disk storage management often requires that machines be taken offline at a major inconvenience to users. In the distributed client/server environment, databases and other resources must maintain high availability, be easy to access, and be safe from damage caused by hardware malfunction.

VERITAS Volume Manager provides the tools to improve performance and ensure data availability and integrity. Volume Manager also *dynamically configures disk storage while the system is active*.

This chapter introduces VERITAS Volume Manager concepts and describes the tools that Volume Manager uses to perform storage management.

The following topics are covered in this chapter:

- How Data is Stored
- Volume Manager Overview
- Physical Objects
- Volumes and Virtual Objects
- Volume Layouts
- Volume Manager and RAID-5
- Layered Volumes
- Volume Manager User Interfaces

How Data is Stored

There are several methods used to store data on physical disks. These methods organize data on the disk so the data can be stored and retrieved efficiently. The basic method of disk organization is called *formatting*. Formatting prepares the hard disk so that files can be written to and retrieved from the disk by using a prearranged storage pattern.

Hard disks are formatted, and information stored, in two ways: physical-storage layout and logical-storage layout. Volume Manager uses the *logical-storage layout* method. The types of storage layout supported by Volume Manager are introduced in this chapter.

Volume Manager Overview

The Volume Manager uses *objects* to do storage management. The two types of objects used by Volume Manager are *physical objects* and *virtual objects*.

- physical objects

  Volume Manager uses physical disksto store data.

- virtual objects

  Volume Manager creates virtual objects called *volumes* on physical disks.. Each volume records and retrieves data from one or more physical disks. Volumes are accessed by file systems, databases, or other applications in the same way that physical disks are accessed. Volumes are also composed of other virtual objects that are used to change the volume configuration. Volumes and their virtual components are called *virtual objects* or Volume Manager objects. Volume Manager objects can be ussed to perform administrative tasks on disks without interrupting applications and users.

Physical Objects

This section describes the physical objects (physical disks) used by Volume Manager.

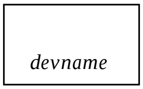
Physical Disks and Disk Naming

A *physical disk* is the basic storage device (media) where the data is ultimately stored. You can access the data on a physical disk by using a *device name* to locate the disk. The physical disk device name varies with the computer system you use. Not all parameters are used on all systems. Typical device names can include: `c#t#d#`, where:

- `c#` is the controller
- `t#` is the target ID
- `d#` is the disk number

Figure 1, "Example of a Physical Disk," shows how a physical disk and device name (*devname*) are illustrated in this document. For example, device name `c0t0d0` is connected to controller number `0` in the system, with a target ID of `0`, and physical disk number `0`.

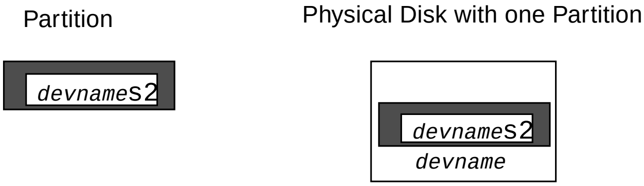
Figure 1. Example of a Physical Disk



Partitions

On some computer systems, a physical disk can be divided into one or more *partitions*. The *partition number*, or `s`#, is added at the end of the devname. Note that a partition can be an entire physical disk, such as the partition shown in Figure 2, "Example of a Partition."

Figure 2. Example of a Partition



Volumes and Virtual Objects

The connection between physical objects and Volume Manager objects is made when you place a physical disk under Volume Manager control.

Volume Manager creates virtual objects (or Volume Manager objects) and makes logical connections between the objects. The virtual objects are then used by Volume Manager to do storage management tasks. Volume Manager objects include:

- disk groups
- VM disks
- volumes
- plexes (mirrors)

- subdisks

These objects are described in later sections.

A volume is a virtual disk device that appears to applications, databases, and file systems as a physical disk. However, a volume does not have the limitations of a physical disk. When you use Volume Manager, applications access volumes created on Volume Manager disks (VM Disks) rather than physical disks.

Volumes contain other virtual objects that you can use to manipulate data within a volume. The virtual objects contained in volumes are subdisks and plexes. Details of the virtual objects are described in the following sections. The combination of virtual objects and how they are manipulated by volumes is described in “Volumes” on page 22.

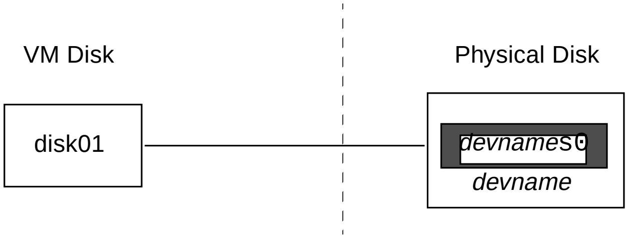
Volume Manager Disks

When you place a physical disk under Volume Manager control, a Volume Manager disk (or VM Disk) is assigned to the physical disk. A VM Disk is under Volume Manager control and is usually in a disk group. Each VM disk corresponds to at least one physical disk.

A VM disk typically includes a *public region* (allocated storage) and a *private region* where Volume Manager internal configuration information is stored.

Each VM Disk has a unique *disk media name* (a virtual disk name). You can supply the disk name or allow Volume Manager to assign a default name that typically takes the form `disk##`. Figure 3, ”Example of a VM Disk,” shows a VM disk with a media name of `disk01` that is assigned to the physical disk *devname*.

Figure 3. Example of a VM Disk



Disk Groups

A *disk group* is a collection of VM disks that share a common configuration. A disk group configuration is a set of records with detailed information about related Volume Manager objects, their attributes, and their connections. The default disk group is `rootdg` (the root disk group).

You can create additional disk groups as necessary. Disk groups allow you to group disks into logical collections. A disk group and its components can be moved as a unit from one host machine to another.

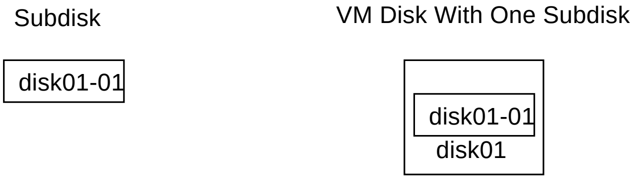
Volumes are created within a disk group. A given volume must be configured from disks in the same disk group.

Subdisks

A *subdisk* is a set of contiguous disk blocks. A block is a unit of space on the disk. Volume Manager allocates disk space using subdisks. 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.

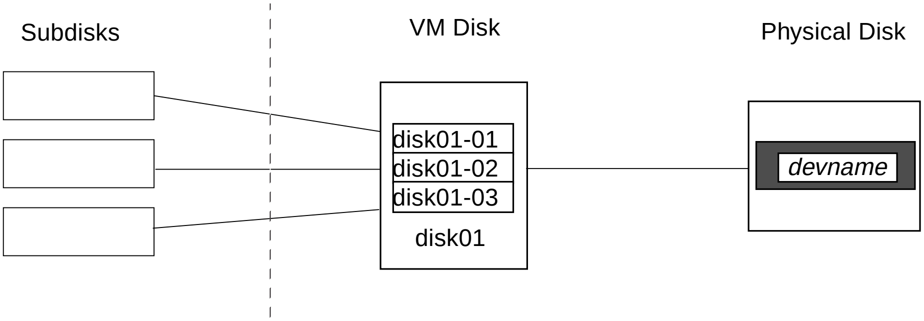
The default name for a VM disk is `disk##` (such as `disk01`) and the default name for a subdisk is `disk##-##`. In Figure 4, "Example of a Subdisk," `disk01-01` is the name of the first subdisk on the VM disk named `disk01`.

Figure 4. Example of a Subdisk



A VM disk can contain multiple subdisks, but subdisks cannot overlap or share the same portions of a VM disk. Figure 5, "Example of Three Subdisks Assigned to One VM Disk," shows a VM disk with three subdisks. The VM disk is assigned to one physical disk.

Figure 5. Example of Three Subdisks Assigned to One VM Disk



Any VM disk space that is not part of a subdisk is free space. You can use free space to create new subdisks.

Volume Manager release 3.0 or higher allows subdisks to contain volumes. For previous versions of Volume Manager, subdisks cannot contain volumes. For more information, see "Layered Volumes" on page 39.

Plexes

The Volume Manager uses subdisks to build virtual objects called *plexes* (or mirrors). A plex consists of one or more subdisks located on one or more physical disks. To organize data on the subdisks to form a plex, use the following methods:

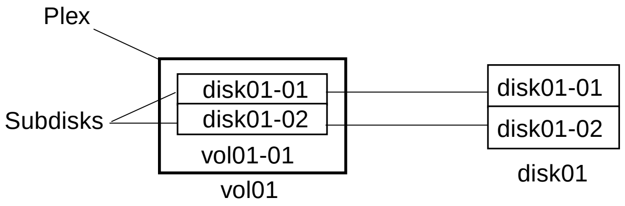
- concatenation
- striping (RAID-0)
- striping with parity (RAID-5)
- mirroring (RAID-1)

Note A Redundant Array of Independent Disks (RAID) is a disk array where part of the combined storage capacity is used to store duplicate information about the data in the array, allowing you to regenerate the data in case of a disk failure.

Figure 6, "Example Plex With Two Subdisks," shows a plex with two subdisks.

Concatenation, striping (RAID-0), RAID-5, and mirroring (RAID-1) are described in "Volume Layouts" on page 25.

Figure 6. Example Plex With Two Subdisks



Volumes

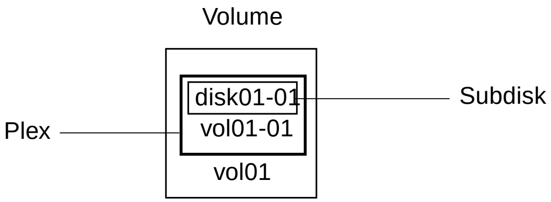
A volume consists of one or more plexes, each holding a copy of the data in the volume. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area of a disk. The configuration of a volume can be changed by using the Volume Manager user interfaces. Configuration changes can be done 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. A volume must have at least one associated plex that has a complete copy of the data in the volume with at least one associated subdisk. Note that all subdisks within a volume must belong to the same disk group.

A volume with two or more data plexes is “mirrored” and contains mirror images of the data. Each plex contains an identical copy of the volume data. Refer to “Mirroring (RAID-1)” on page 33 for more information on mirrored volumes.

The Volume Manager uses the default naming conventions of `vol`*##* for volumes and `vol`*##*-*##* for plexes in a volume. You should select meaningful names for your volumes. Figure 7, ”Example of a Volume with One Plex,” shows a volume with a single plex.

Figure 7. Example of a Volume with One Plex

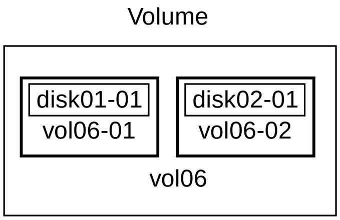


Volume `vol01` in Figure 7 has the following characteristics:

- 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`

Figure 8, ”Example of a Volume with Two Plexes,” shows a mirrored volume with two plexes.

Figure 8. Example of a Volume with Two Plexes



Volume vol06 in Figure 8 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)

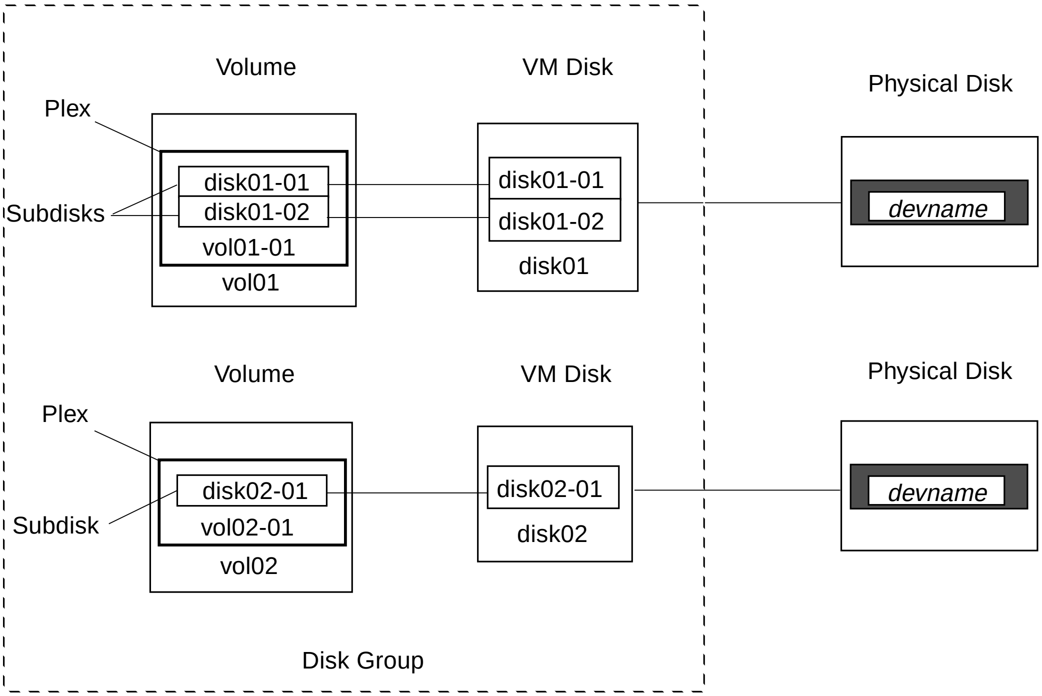
Connection Between Volume Manager Virtual Objects

Volume Manager virtual objects are combined to build volumes. The virtual objects contained in volumes are: VM disks, disk groups, subdisks, and plexes. Volume Manager objects have the following connections:

- Volume Manager disks are grouped into disk groups
- one or more subdisks (each representing a specific region of a disk) are combined to form plexes
- a volume is composed of one or more plexes

The example in Figure 9, ”Connection Between Volume Manager Objects,” shows the connections between Volume Manager virtual objects and how they relate to physical disks. Figure 9 shows a disk group with two VM disks (disk01 and disk02). disk01 has a volume with one plex and two subdisks. disk02 has a volume with one plex and a single subdisk.

Figure 9. Connection Between Volume Manager Objects



Volume Layouts

You can organize data in virtual objects to create volumes by using these layout methods:

- concatenation
- striping (RAID-0)
- RAID-5 (striping with parity)
- mirroring (RAID-1)
- mirroring plus striping
- striping plus mirroring

The following sections describe each layout method.

A Volume Manager virtual device is defined by a volume. A volume has a layout defined by the association of a volume to one or more plexes, which in turn, each map to subdisks. The volume then presents a virtual device interface exposed to Volume Manager clients for data access. These logical building blocks re-map the volume address space through which I/O is re-directed at run-time.

Different volume layouts each provide different levels of storage service. A volume layout can be configured and reconfigured to match particular levels of desired storage service.

In previous releases of Volume Manager, the subdisk was restricted to mapping directly to a VM disk. This allowed the subdisk to define a contiguous extent of storage space backed by the public region of a VM disk. When active, the VM disk is associated with an underlying physical disk, this is how Volume Manager logical objects map to physical objects, and stores data on stable storage.

The combination of a volume layout and the physical disks which provide backing store, therefore determine the storage service available from a given virtual device.

In the 3.0 or higher releases of Volume Manager “layered volumes” can be constructed by permitting the subdisk to map either to a VM disk as before, or, to a new logical object called a *storage volume*. A storage volume provides a recursive level of mapping with layouts similar to the top-level volume. Eventually, the “bottom” of the mapping requires an association to a VM disk, and hence to attached physical storage.

Layered volumes allow for more combinations of logical compositions, some of which may be desirable for configuring a virtual device. Because permitting free use of layered volumes throughout the command level would have resulted in unwieldy administration, some ready-made layered volume configurations were designed into the Volume Manager.

These ready-made configurations operate with built-in rules to automatically match desired levels of service within specified constraints. The automatic configuration is done on a “best-effort” basis for the current command invocation working against the current configuration.

To achieve the desired storage service from a set of virtual devices, it may be necessary to include an appropriate set of VM disks into a disk group, and to execute multiple configuration commands.

To the extent that it can, Volume Manager handles initial configuration and on-line re-configuration with its set of layouts and administration interface to make this job easier and more deterministic.

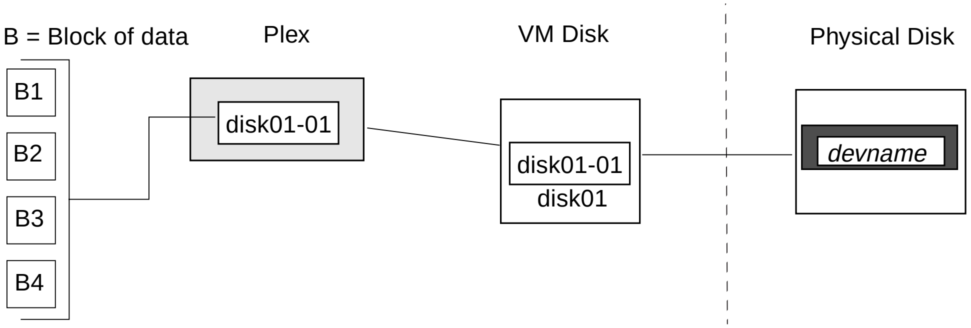
Concatenation

Concatenation maps data in a linear manner onto one or more subdisks in a plex. To access all the data in a concatenated plex sequentially, data is first accessed in the first subdisk from beginning to end. Data is then accessed in the remaining subdisks sequentially from beginning to end until the end of the last subdisk.

The subdisks in a concatenated plex do not 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 called *spanning*.

Figure 10, ”Example of Concatenation,” shows concatenation with one subdisk.

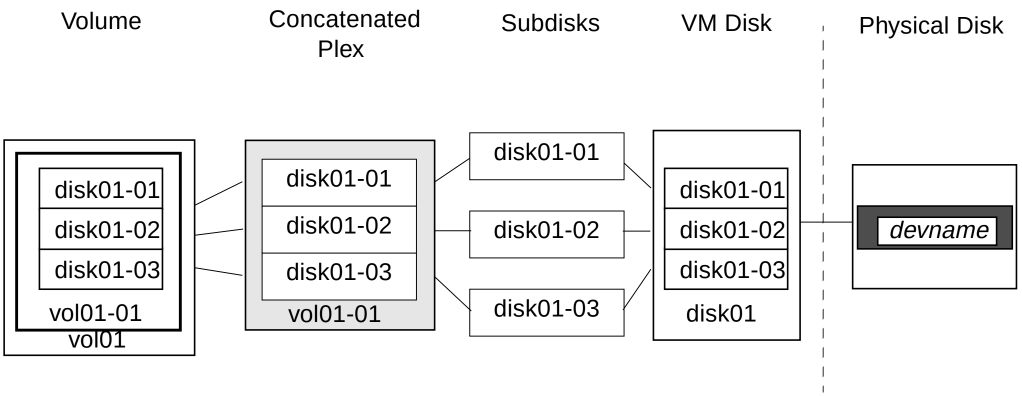
Figure 10. Example of Concatenation



You can use concatenation with multiple subdisks when there is insufficient contiguous space for the plex on any one disk. This form of concatenation can be used for load balancing between disks, and for head movement optimization on a particular disk.

Figure 11, ”Example of a Volume in a Concatenated Configuration,” shows a volume in a concatenated configuration.

Figure 11. Example of a Volume in a Concatenated Configuration

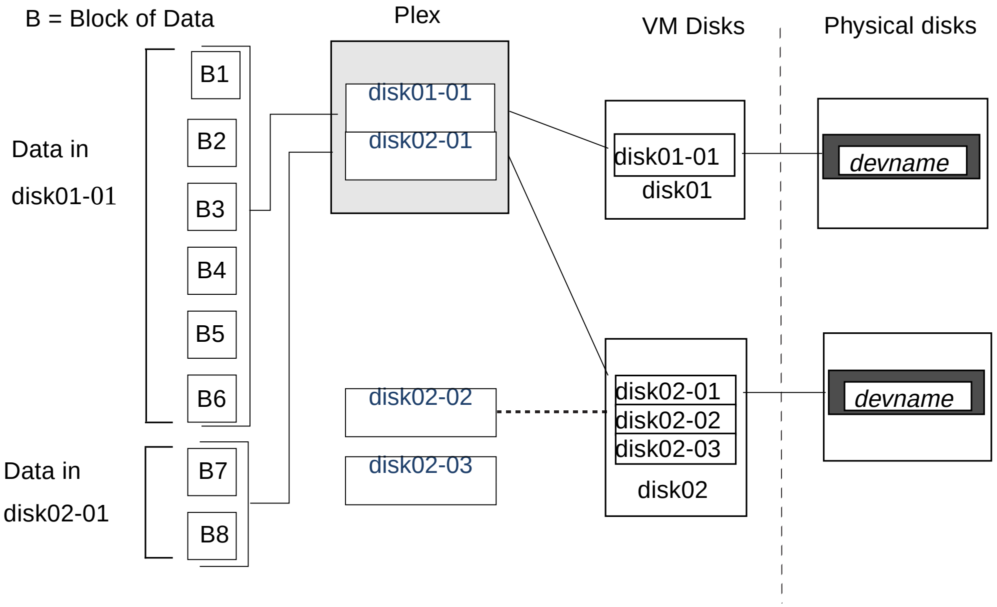


In the example shown in Figure 12, the first six blocks of data (B1 through B6) use most of the space on the disk that VM disk `disk01` is assigned to. This requires space only on subdisk `disk01-01` on VM disk `disk01`. However, the last two blocks of data, B7 and B8, use only a portion of the space on the disk that VM disk `disk02` is assigned to.

The remaining free space on VM disk `disk02` can be put to other uses. In this example, subdisks `disk02-02` and `disk02-03` are available for other disk management tasks.

Figure 12, "Example of Spanning," shows data spread over two subdisks in a spanned plex.

Figure 12. Example of Spanning



Caution Spanning a plex across multiple disks increases the chance that a disk failure results in failure of the assigned volume. Use mirroring or RAID-5 (both described later) to reduce the risk that a single disk failure results in a volume failure.

Striping (RAID-0)

Striping (RAID-0) maps data so that the data is interleaved among two or more physical disks. 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.

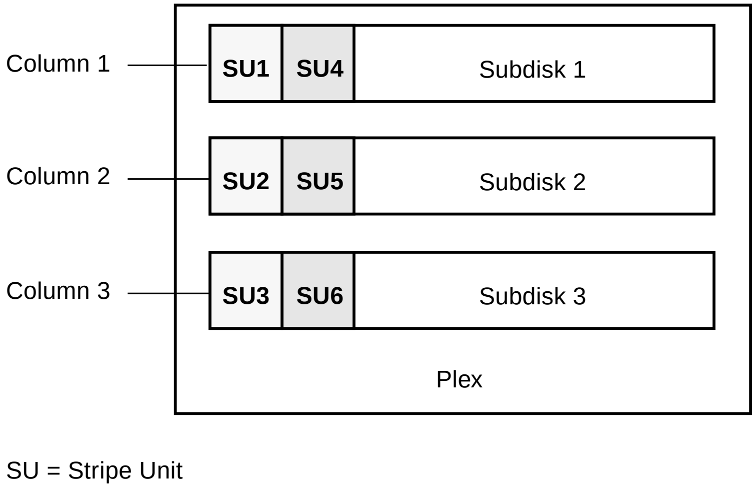
Caution Striping a volume, or splitting a volume across multiple disks, increases the chance that a disk failure will result in failure of that volume. For example, if five volumes are striped across the same five disks, then failure of any one of the five disks will require that all five volumes be restored from a backup. If each volume is on a separate disk, only one volume has to be restored. Use mirroring or RAID-5 to substantially reduce the chance that a single disk failure results in failure of a large number of volumes.

Data is allocated in equal-sized units (*stripe 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 shown in Figure 13, "Striping Across Three Disks (Columns)":

- the first and fourth stripe units are allocated in column 1
- the second and fifth stripe units are allocated in column 2
- the third and sixth stripe units are allocated in column 3

Figure 13. Striping Across Three Disks (Columns)



A *stripe* consists of the set of stripe units at the same positions across all columns. In Figure 13, stripe units 1, 2, and 3 constitute a single stripe.

Viewed in sequence, the first stripe consists of:

- stripe unit 1 in column 1
- stripe unit 2 in column 2
- stripe unit 3 in column 3

The second stripe consists of:

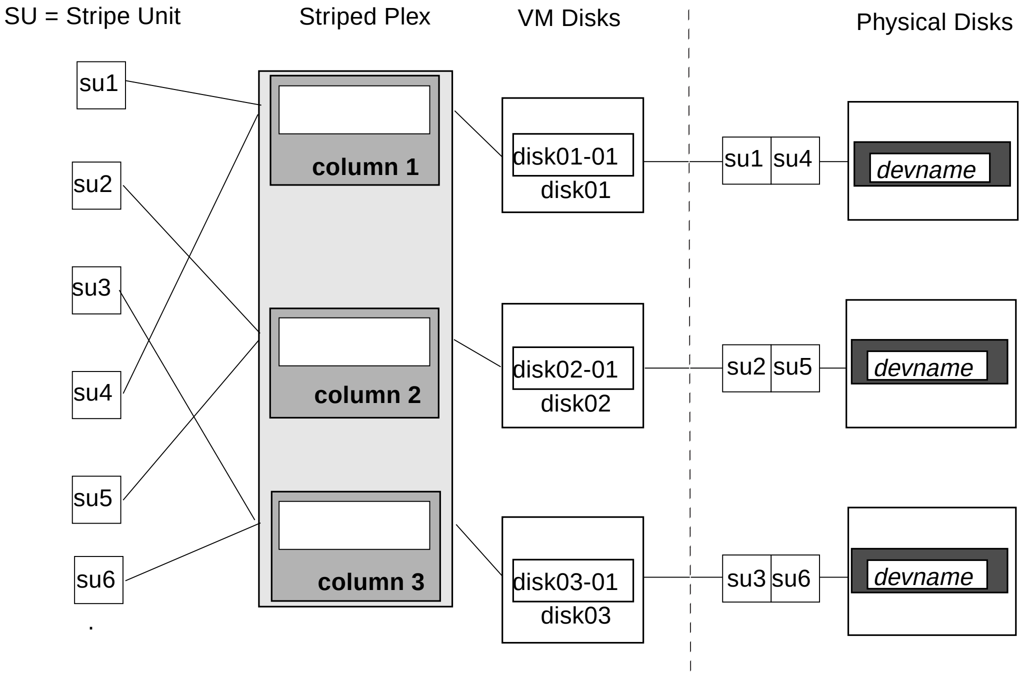
- stripe unit 4 in column 1
- stripe unit 5 in column 2
- stripe unit 6 in column 3

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.

Striping is useful if you need large amounts of data 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.

Figure 14, "Example of a Striped Plex with One Subdisk per Column," shows a striped plex with three equal sized, single-subdisk columns. There is one column per physical disk.

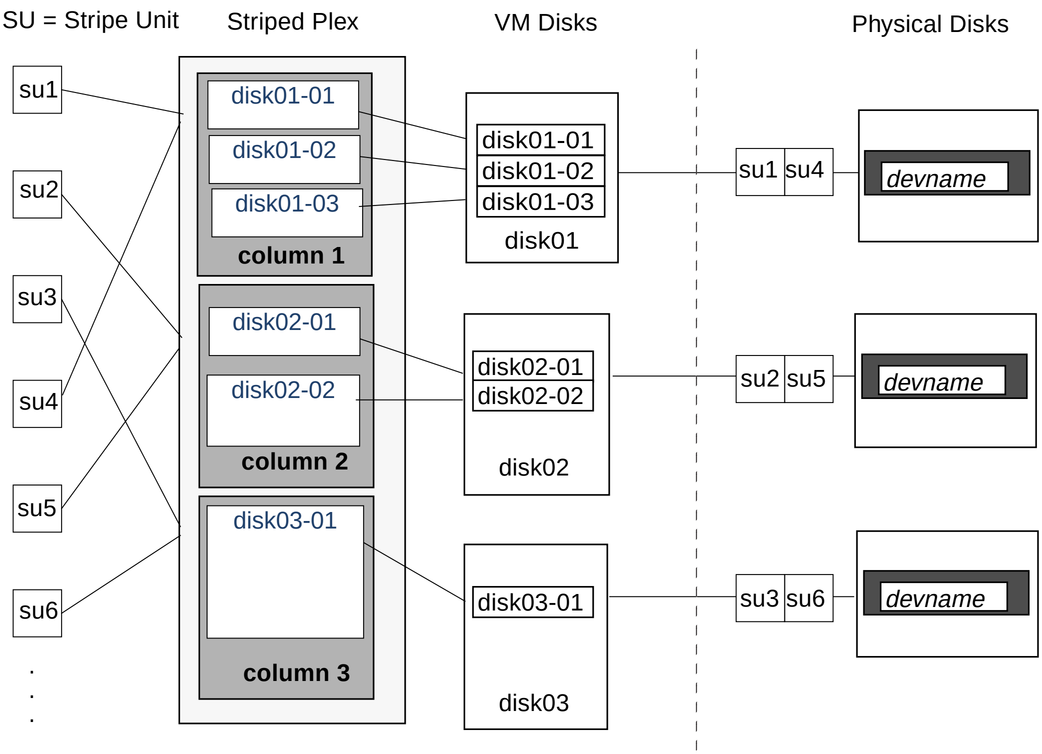
Figure 14. Example of a Striped Plex with One Subdisk per Column



The example in Figure 14 shows three subdisks that occupy all of the space on the VM disks. It is also possible for each subdisk in a striped plex to occupy only a portion of the VM disk, which leaves free space for other disk management tasks.

Figure 15, ”Example of a Striped Plex with Concatenated Subdisks per Column,” shows a striped plex with three columns containing subdisks of different sizes. Each column contains a different number of subdisks. There is one column per physical disk. Striped plexes can be created by 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 (for example, if the plex is grown). Columns can contain subdisks from different VM disks, also.

Figure 15. Example of a Striped Plex with Concatenated Subdisks per Column



RAID-5

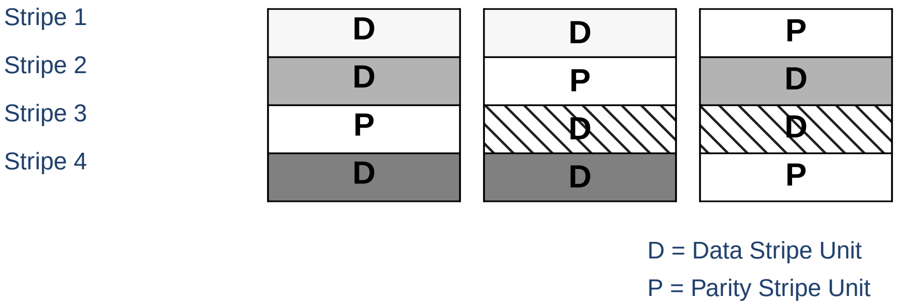
RAID-5 provides data redundancy by using *parity.* Parity is a calculated value used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is calculated by doing an *exclusive OR* (XOR) procedure on the data. The resulting parity is then written to the volume. If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

RAID-5 volumes maintain redundancy of the data within a volume. RAID-5 volumes keep a copy of the data and calculated parity in a plex that is “striped” across multiple disks. In the event of a disk failure, a RAID-5 volume uses parity to reconstruct the data. It is possible to mix concatenation and striping in the layout.

RAID-5 volumes can do logging to minimize recovery time. RAID-5 volumes use RAID-5 logs to keep a copy of the data and parity currently being written. RAID-5 logging is optional and can be created along with RAID-5 volumes or added later.

Figure 16, ”Parity Locations in a RAID-5 Model,” shows 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.

Figure 16. Parity Locations in a RAID-5 Model



For more information, see “Volume Manager and RAID-5” on page 35.

Mirroring (RAID-1)

Mirroring uses multiple mirrors (plexes) to duplicate the information contained in a volume. In the event of a physical disk failure, the plex 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 must contain disk space from different disks to achieve redundancy.

When striping or spanning across a large number of disks, failure of any one of those disks can 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.

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

The Volume Manager supports the combination of mirroring plus striping. The combined layout is called mirror-stripe layout. When used together on the same volume, mirroring plus striping offers the benefits of spreading data across multiple disks (striping) while providing redundancy (mirror) of data.

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

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

The Volume Manager supports the combination of striping with mirroring. This combined layout is called stripe-mirror layout. Now there can be mirroring both above and below striping.

Putting mirroring below striping mirrors each column of the stripe. If the stripe is large enough to have multiple subdisks per column, each subdisk can be individually mirrored. This layout enhances redundancy and reduces recovery time in case of an error.

If a disk fails in a mirror- stripe layout, the entire plex is detached, thereby losing redundancy on the entire volume. When the disk is replaced, the entire plex must be brought up to date. Recovering the entire plex can take a substantial amount of time. If a disk fails in a stripe-mirror layout, only the failing subdisk must be detached, and only that portion of the volume loses redundancy. When the disk is replaced, only a portion of the volume needs to be recovered.

Compared to mirroring plus striping, striping plus mirroring offers a volume more tolerant to disk failure. If a disk failure occurs, the recovery time is shorter for striping plus mirroring. See “Layered Volumes” on page 39 for more information.

Volume Manager and RAID-5

This section describes how Volume Manager implements RAID-5.

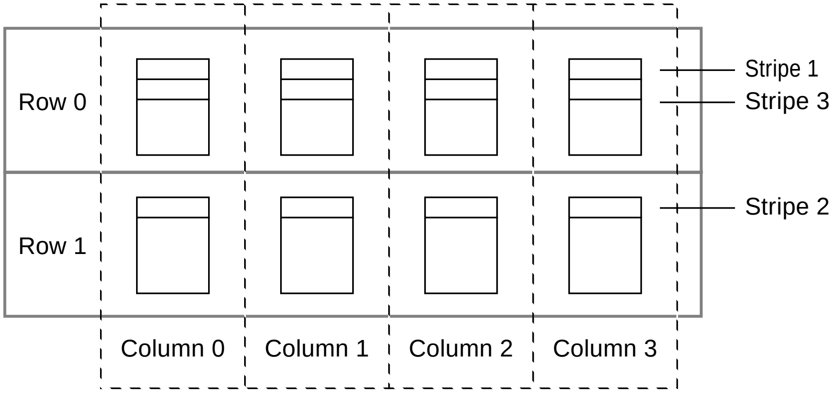
Although both mirroring (RAID-1) and RAID-5 provide redundancy of data, they use different methods. Mirroring provides data redundancy by maintaining multiple complete copies of the data in a volume. Data being written to a mirrored volume is reflected in all copies. If a portion of a mirrored volume fails, the system continues to use the other copies of the data.

RAID-5 provides data redundancy by using *parity*. Parity is a calculated value used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is calculated by doing an exclusive OR (XOR) procedure on the 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 recreated from the remaining data and parity information.

Traditional RAID-5 Arrays

A *traditional* RAID-5 array is several disks organized in rows and columns. A *column* is a number of disks located in the same ordinal position in the array. A *row* is the minimal number of disks necessary to support the full width of a parity stripe. Figure 17, ”Traditional RAID-5 Array,” shows the row and column arrangement of a traditional RAID-5 array.

Figure 17. Traditional RAID-5 Array

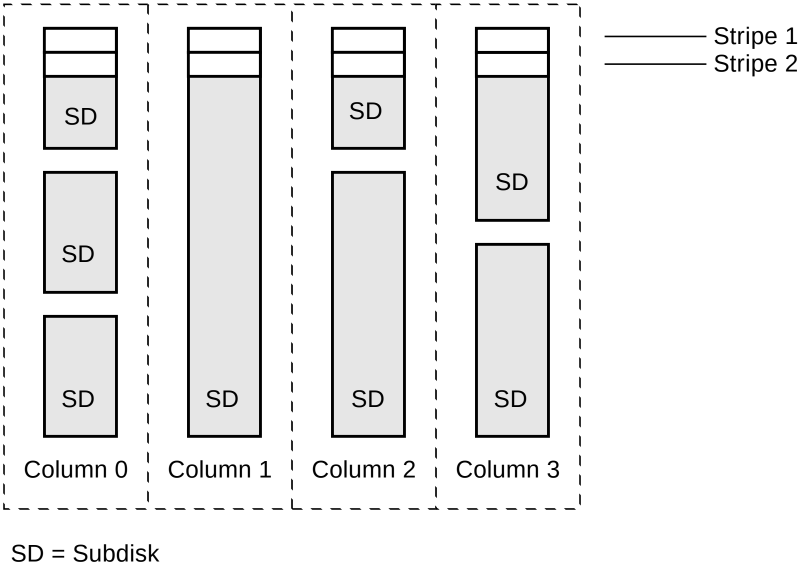


This traditional array structure supports growth by adding more rows per column. Striping is accomplished by applying the first stripe across the disks in Row 0, then the second stripe across the disks in Row 1, then the third stripe across the Row 0 disks, and so on. This type of array requires all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

Volume Manager RAID-5 Arrays

The Volume Manager 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 use rows. Instead, the Volume Manager uses columns consisting of variable length subdisks (as shown in Figure 18, ”Volume Manager RAID-5 Array”). Each subdisk represents a specific area of a disk.

Figure 18. Volume Manager RAID-5 Array



With the Volume Manager RAID-5 array structure, each column can consist of a different number of subdisks. The subdisks in a given column can be derived from different physical disks. Additional subdisks can be added to the columns as necessary. Striping (see “Striping (RAID-0)” on page 29) is done by applying the first stripe across each subdisk at the top of each column, then another stripe below that, and so on for the 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.

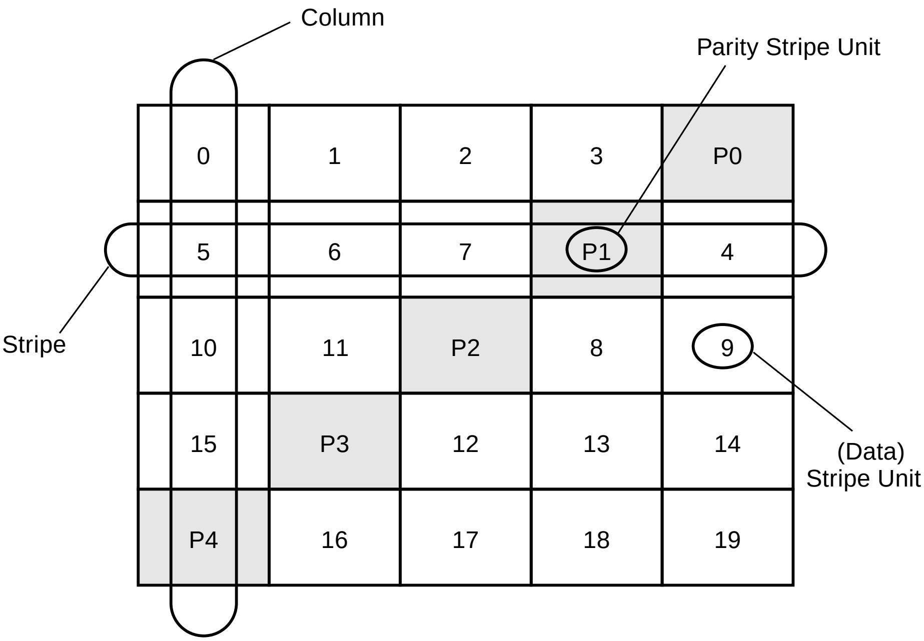
Left-Symmetric Layout

There are several layouts for data and parity that can be used in the setup of a RAID-5 array. The Volume Manager implementation of RAID-5 is the left-symmetric layout. The left-symmetric parity layout provides optimal performance for both random I/O operations and large sequential I/O operations. 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, shifted left 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 19, "Left-Symmetric Layout," shows a left-symmetric parity layout with five disks (one per column).

Figure 19. Left-Symmetric Layout



0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

For each stripe, data is organized starting to the right of the parity stripe unit. In Figure 19, 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 done 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. Data is 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 far left column in Figure 19 fails, the volume is placed in a degraded mode. While in degraded mode, the data from the failed column can be recreated by XORing stripe units 1-3 against parity stripe unit P0 to recreate 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 detaches the volume. The volume is no longer allowed to satisfy read or write requests. Once the failed columns have been recovered, it may be necessary to recover the user data from backups.

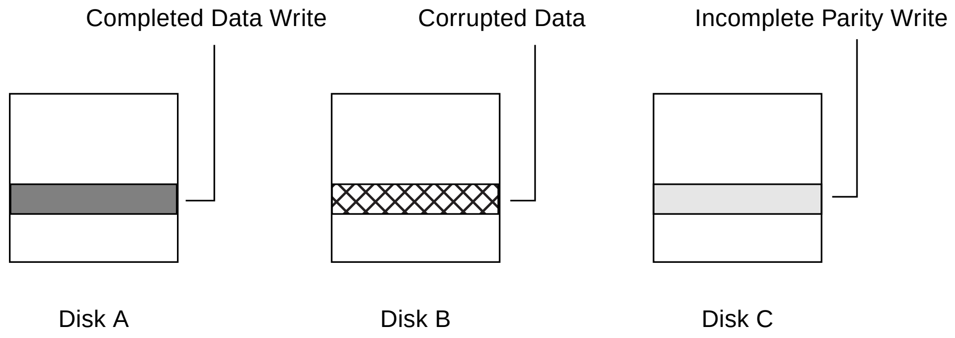
Logging

Logging (recording) is used to prevent 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.

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.

In Figure 20, ”Incomplete Write,” the recovery of Disk B is dependent on the data on Disk A and the parity on Disk C having both been 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 20. Incomplete Write



This failure case can be avoided 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 log plexes. More than one log plex can exist for each RAID-5 volume, in which case the log areas are mirrored.

Layered Volumes

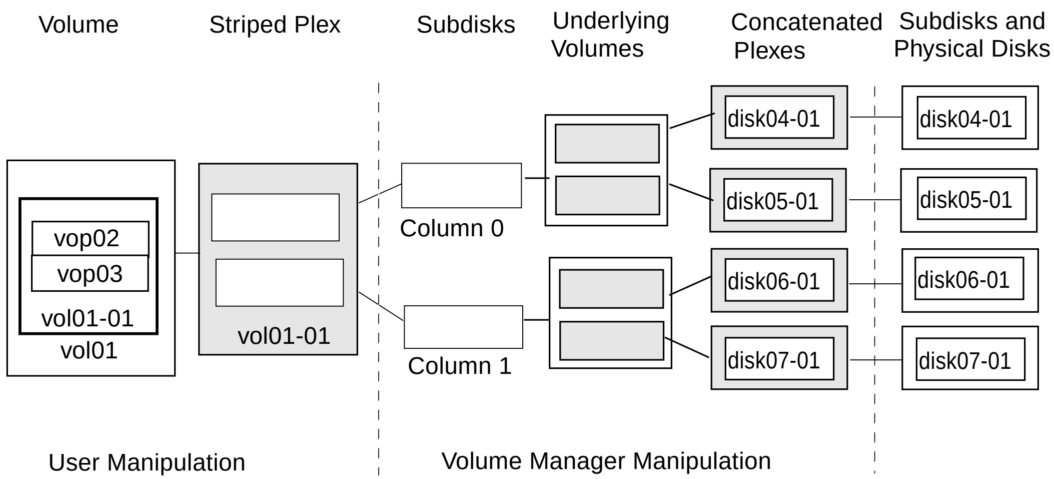
A layered volume is a virtual Volume Manager object that is built on top of volume(s). The layered volume structure tolerates failure better and has greater redundancy than the standard volume structure. For example, in a striped and mirrored layered volume, each mirror (plex) covers a smaller area of storage space, so recovery is quicker than with a standard mirrored volume. Figure 21, ”Example of a Striped-Mirrored Layered Volume,” shows an example of a layered volume.

Note Volume Manager 3.0, and later, supports layered volumes but previous versions do not support layered volumes. Volume Manager 3.0 or higher allows subdisks to be built on volumes (storage volumes). but previous versions of Volume Manager do not support subdisks.

User tasks can be performed only on the top-level volume of a layered volume. You cannot detach a layered volume or perform any other operation on the underlying volumes by manipulating the internal structure. You can perform all necessary operations from the user manipulation area that includes the volume and striped plex. In Figure 21, the volume and striped plex in the “User Manipulation” area allow you to perform normal Volume Manager tasks.

The “Volume Manager Manipulation” area in Figure 21 shows subdisks with two columns, built on underlying volumes with each volume internally mirrored. Layered volumes are an infrastructure within Volume Manager and they allow the addition of certain features to be added to Volume Manager. Underlying volumes are used exclusively by the Volume Manager and are not designed for user manipulation. The underlying volume structure is described here to help you understand how layered volumes work and why they are used by Volume Manager.

Figure 21. Example of a Striped-Mirrored Layered Volume



System administrators may need to manipulate the layered volume structure for troubleshooting or other operations (for example, to place data on specific disks). Layered volumes are used by Volume Manager to perform these tasks and operations:

- striped-mirrors (see the `vxassist` manual page)
- concatenated mirrors (see the `vxassist` manual page)
- Online Relayout (see the `vxrelayout` and `vxassist` manual pages)
- RAID-5 subdisk moves (see the `vxsd` manual page)
- RAID-5 snapshot (see the `vxassist` manual page)

Volume Manager and the Operating System

Volume Manager operates as a subsystem between your operating system and your data management systems, such as file systems and database management systems.

Before a disk can be brought under Volume Manager control, the disk must be accessible through the operating system device interface. Volume Manager is a subsystem layered on top of the operating system interface services. Therefore, Volume Manager is dependent upon how the operating system accesses physical disks.

Volume Manager is dependent upon the operating system for the following.

- operating system (disk) devices
- device handles
- VM disks
- Volume Manager dynamic multipathing (DMP) metadevice

Volume Manager Layouts

A Volume Manager virtual device is defined by a volume. A volume has a layout defined by the association of a volume to one or more plexes, which in turn, each map to subdisks. The volume then presents a virtual device interface exposed to Volume Manager clients for data access. These logical building blocks re-map the volume address space through which I/O is re-directed at run-time.

Different volume layouts each provide different levels of storage service. A volume layout can be configured and re-configured to match particular levels of desired storage service.

In previous releases of Volume Manager, the subdisk was restricted to mapping directly to a VM disk. This allowed the subdisk to define a contiguous extent of storage space backed by the public region of a VM disk. When active, the VM disk is associated with an underlying physical disk, this is how Volume Manager logical objects map to physical objects, and stores data on stable storage.

The combination of a volume layout and the physical disks which provide backing store, therefore determine the storage service available from a given virtual device.

In the 3.0 or higher releases of Volume Manager “layered volumes” can be constructed by permitting the subdisk to map either to a VM disk as before, or, to a new logical object called a *storage volume*. A storage volume provides a recursive level of mapping with layouts similar to the top-level volume. Eventually, the “bottom” of the mapping requires an association to a VM disk, and hence to attached physical storage.

Layered volumes allow for more combinations of logical compositions, some of which may be desirable for configuring a virtual device. Because permitting free use of layered volumes throughout the command level would have resulted in unwieldy administration, some ready-made layered volume configurations have been designed into Volume Manager.

These ready-made configurations operate with built-in rules to automatically match desired levels of service within specified constraints. The automatic configuration is done on a “best-effort” basis for the current command invocation working against the current configuration.

To achieve the desired storage service from a set of virtual devices, it may be necessary to include an appropriate set of VM disks into a disk group, and to execute multiple configuration commands.

To the extent that it can, the 3.0 release of Volume Manager handles initial configuration and on-line re-configuration with its set of layouts and administration interface to make this job easier and more deterministic.

Volume Manager User Interfaces

This section briefly describes the VERITAS Volume Manager user interfaces.

User Interface Overview

The Volume Manager supports the following user interfaces:

- Volume Manager Storage Administrator (VMSA)

  The Storage Administrator is a graphical user interface to the Volume Manager. The Storage Administrator provides visual elements such as icons, menus, and dialog boxes to manipulate Volume Manager objects. The Storage Administrator also acts as an interface to some common file system operations. The Storage Administrator is described in the *VERITAS Volume Manager Storage Administrator Administrator’s Guide*.

- Command Line Interface

  Volume Manager commands range from simple commands to complex commands requiring detailed user input. Many Volume Manager commands require you to have an understanding of Volume Manager concepts. Volume Manager concepts are described in this chapter. Most Volume Manager commands require superuser or other appropriate privileges. The command line interface is described in this manual..

- Volume Manager Support Operations

  Volume Manager Support Operations interface (`vxdiskadm`) is a menu-driven interface for disk and volume administration functions. `vxdiskadm` uses a main menu where you can select storage management tasks to be performed. `vxdiskadm` is described in the *VERITAS Volume Manager Reference Guide*.

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

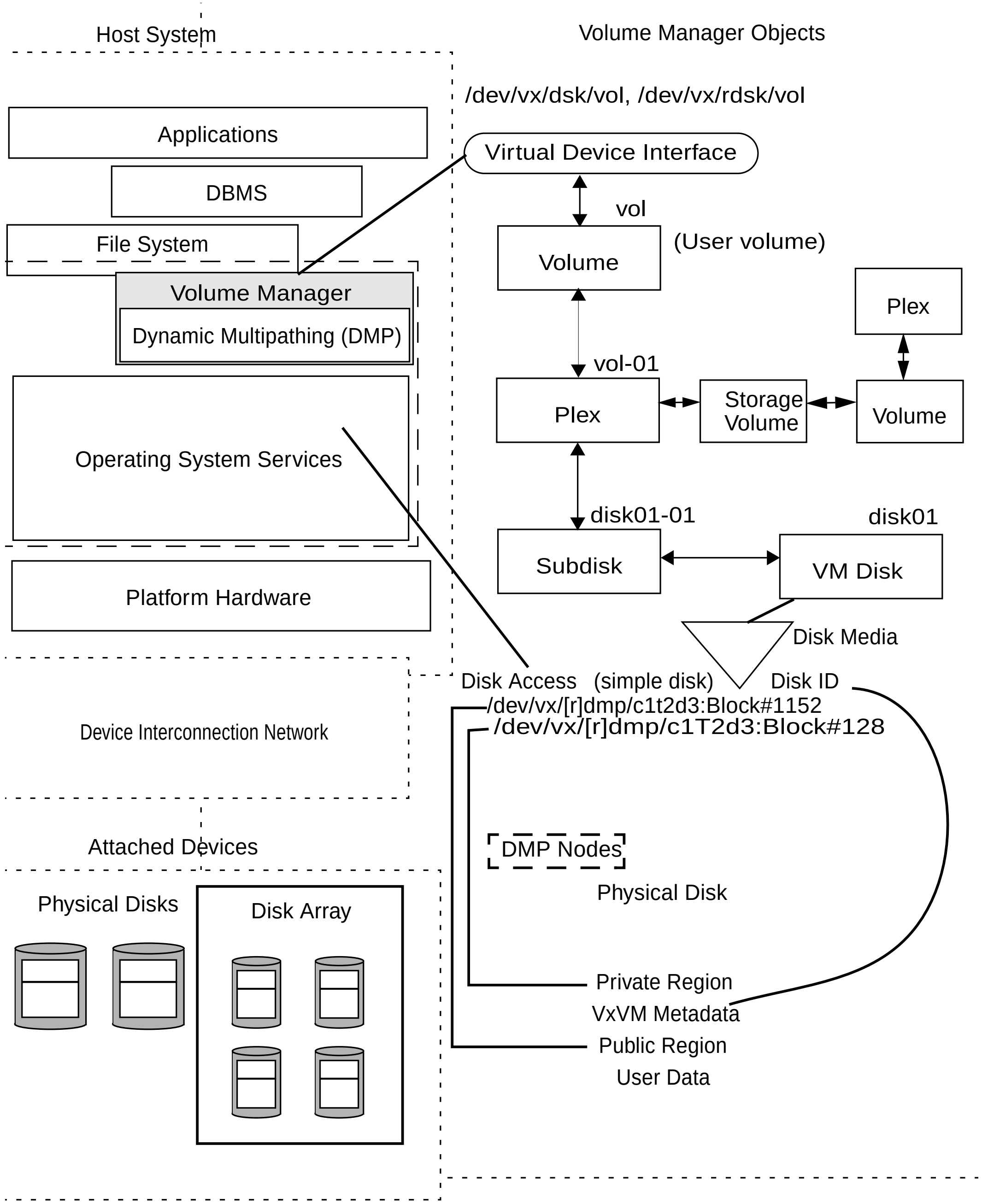
Why You Should Use Volume Manager

Volume Manager provides enhanced data storage service by separating the physical and logical aspects of data management. Volume Manager enhances data storage by controlling these aspects of storage:

- space—allocation and use
- performance— enhanced data delivery
- data availability—continuous operation and multisystem access
- device installation—centralized and optimized support
- system—multisystem support and monitoring of private/shared systems

See Figure 22, ”Volume Manager System Concepts,” on page 44.

Figure 22. Volume Manager System Concepts



After installing Volume Manager on a host system, perform the following procedure before you can configure and use Volume Manager objects:

- bring the contents of physical disks under Volume Manager control
- collect the Volume Manager disks into disk groups
- allocate the disk group space to create logical volumes

Bringing the contents of physical disks under Volume Manager control is done only if:

- you allow Volume Manager to take control of the physical disks
- the disk is not under control of another storage manager

Volume Manager writes identification information on physical disks under Volume Manager control (claimed disks). Claimed disks can be identified even after physical disk disconnection or system outages. Volume Manager can then re-form disk groups and logical objects to provide failure detection and to speed system recovery.

Getting Started 2

Introduction

This section briefly describes the steps needed to setup Volume Manager and the daemons that must be running for Volume Manager to operate. This chapter also provides guidelines to help you set up a system with storage management.

Refer to the *VERITAS Volume Manager Installation Guide* for detailed information on how to install and set up the Volume Manager and the Storage Administrator.

The following topics are covered in this chapter:

- Volume Manager Initialization
- Volume Manager Daemons
- System Setup
- System Setup Guidelines
- Protecting Your System

Volume Manager Initialization

You initialize the Volume Manager by using the `vxinstall` program. `vxinstall` places specified disks under Volume Manager control. By default, these disks are placed in the `rootdg` disk group. You must use `vxinstall` to initialize at least one disk into `rootdg.` You can then use `vxdiskadm` or the Storage Administrator to initialize or encapsulate additional disks into disk groups.

Once you have completed the package installation, follow these steps to initialize the Volume Manager.

1. Log in as superuser.

2. Create a `disks.exclude` file if you want to exclude disks from Volume Manager control. `vxinstall` ignores any disks listed in this file. Place this file in: `/etc/vx/disks.exclude.`

3. Create a `cntrls.exclude` file if you want to exclude all disks on a controller from Volume Manager control. Place this file in: `/etc/vx/cntrls.exclude`.

4. Start `vxinstall` by entering this command: `vxinstall.`

`vxinstall` then does the following:

- runs and displays the license information and prompts for a key
- examines and lists all controllers attached to the system
- allows you to choose an initialization process: Quick or Custom installation

Quick installation gives you the option of initializing or encapsulating all disks. To encapsulate some disks on a given controller and initialize others, use the Custom installation process.

Custom installation allows you to control which disks are added to Volume Manager control and how they are added. You can initialize or encapsulate all disks on a controller, or initialize some disks on a controller and encapsulate others.

For details on how to use the Quick or Custom installation option, refer to the *VERITAS Volume Manager Installation Guide.*

The way you choose to configure your system determines whether or not a shutdown and reboot is required. If you choose to encapsulate any disks, a reboot is necessary. `vxinstall` informs you if a reboot is required.

You can use this command to confirm that key Volume Manager processes are running (`vxconfigd`, `vxnotify`, and `vxrelocd`) after you install and initialize VM.

```
# ps -ef | grep vx
```

Volume Manager Daemons

Two daemons must be running for the Volume Manager to operate properly:

- `vxconfigd`
- `vxiod`

Configuration Daemon vxconfigd

The Volume Manager configuration daemon (`vxconfigd`) maintains Volume Manager disk and disk group configurations. `vxconfigd` communicates configuration changes to the kernel and modifies configuration information stored on disk.

Starting the Volume Manager Configuration Daemon

vxconfigd is invoked by startup scripts during the boot procedure.

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

```
# vxdctl mode
```

This message is displayed if vxconfigd is running and enabled:

```
mode: enabled
```

This message is displayed if vxconfigd is running, but not enabled:

```
mode: disabled
```

To enable the volume daemon, enter this command:

```
# vxdctl enable
```

This message is displayed if vxconfigd is not running:

```
mode: not-running
```

To start vxconfigd enter this command:

```
# vxconfigd
```

Once started, vxconfigd automatically becomes a background process.

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

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

Volume I/O Daemon vxiod

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

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

Starting 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 starts vxiod.

Verify that vxiod daemons are running by entering this command:

```
# vxiod
```

Because vxiod is a kernel thread and is not visible to you through the ps command, this is the only way to see if any vxiod daemons are running.

If any vxiod daemons are running, the following message is 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 entering this command:

```
# vxiod set 10
```

where 10 can be substituted by the desired number of vxiod daemons. It is recommended that at least one vxiod daemon exist for each CPU in the system.

System Setup

This section has information to help you set up your system for efficient storage management. For specific setup tasks, refer to other sections of this guide and to the *VERITAS Volume Manager Storage Administrator Administrator's Guide*.

The following system setup sequence is typical and can be used as an example. Your system requirements may differ. The system setup guidelines provide helpful information for specific setup configurations.

Example System Setup Sequence

The following list describes typical steps you may use when setting up your storage management system.

Initial Setup

- Place disks under Volume Manager control.
- Create new disk groups (if you do not want to use rootdg or you want other disk groups).
- Create volumes.
- Put file system(s) on volumes.

Options

- Encapsulate the boot / root disk and mirror it to create alternate boot disk.
- Designate hot-relocation spare disks.
- Add mirrors to volumes.

Maintenance

- Resize volumes and file systems.
- Add more disks / disk groups.
- Create snapshots.

System Setup Guidelines

These general guidelines can help you to understand and plan an efficient storage management system. You can use the cross references in each section to get more information about the featured guideline.

Hot-Relocation Guidelines

You can follow these general guidelines when using hot-relocation. See “Hot-Relocation” on page 64 for more information.

- The hot-relocation feature is enabled by default. Although it is possible to disable hot-relocation, it is advisable to leave it enabled.
- Although hot-relocation does not require you to designate disks as spares, you can 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, Volume Manager uses any available free space within the disk group. When free space is used for relocation purposes, it is possible to have performance degradation after the relocation.
- After hot-relocation occurs, you can designate one or more additional disks as spares to augment the spare space (some of the original spare space may be occupied by relocated subdisks).
- If a given disk group spans multiple controllers and has more than one spare disk, you can set up the spare disks on different controllers (in case one of the controllers fails).
- For a mirrored volume, the disk group must have at least one disk that does not already contain a mirror of the volume. This disk should either be a spare disk with some available space or a regular disk with some free space and the disk is not excluded from hot-relocation use.
- For a mirrored and striped volume, the disk group must have at least one disk that does not already contain one of the mirrors of the volume 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 and the disk is not excluded from hot-relocation use.

- For a RAID-5 volume, the disk group must have at least one disk that does not already contain the RAID-5 plex (or one of its log plexes) of the volume. This disk should either be a spare disk with some available space or a regular disk with some free space and the disk is not excluded from hot-relocation use.
- If a mirrored volume has a DRL log subdisk as part of its data plex, that plex cannot be relocated. You can place log subdisks in plexes that contain no data (log plexes).
- Hot-relocation does not guarantee that it preserves the original performance characteristics or data layout. You can 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 Storage Administrator interface), it is preferable to use spare disks for hot-relocation only.

Striping Guidelines

You can follow these general guidelines when using striping. See “Striping (RAID-0)” on page 29 for more information.

- Do not place more than one column of a striped plex on the same physical disk.
- Calculate stripe unit sizes carefully. In general, a moderate stripe unit size (for example, 64 kilobytes, which is also the default used by `vxassist`) is recommended. 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, it is recommended that you 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 (i.e., 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 these drives, use a moderate stripe unit size (such as 64 kilobytes), unless you know the I/O pattern of the application.

- Volumes with small stripe unit sizes can 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 multi-threaded, 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 rare).

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

Mirroring Guidelines

You can follow these general guidelines when using mirroring. See "Mirroring (RAID-1)" on page 33 for more information.

- Do not place subdisks from different plexes of a mirrored volume on the same physical disk. This action compromises the availability benefits of mirroring and 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 read operations. A higher percentage of read operations results in a higher benefit of performance. Mirroring may not provide a performance increase or result in performance decrease in a write-intensive workload environment.

Note The UNIX operating system implements a file system cache. Read requests can frequently be satisfied from the cache. This can cause the read/write ratio for physical I/O operations through the file system to be 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. Most disk controllers support overlapped seeks that allow 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 important for older controllers or SCSI disks that do not cache on the drive. It is less important for many newer SCSI disks and controllers used in most modern workstations and server machines. Mirroring across controllers can be of benefit because the system can survive a controller failure. The other controller can continue to provide data from the other mirror.

- A plex can exhibit superior performance due to being striped or concatenated across multiple disks, or because it is located on a much faster device. The read policy can then be set to prefer the “faster” plex. By default, a volume with one striped plex is configured with preferred reading of the striped plex.

Dirty Region Logging (DRL) Guidelines

You can follow these general guidelines when using dirty region logging. See “Dirty Region Logging” on page 68 for more information.

Dirty Region Logging (DRL) can speed up recovery of mirrored volumes following a system crash. When DRL is enabled, Volume Manager keeps track of the regions within a volume that have changed as a result of writes to a plex. Volume Manager maintains a bitmap and stores 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, are ignored by plex policies, and are only used to hold the DRL information.

Note Using Dirty Region Logging can impact system performance in a write-intensive environment.

Follow these guidelines when using DRL:

- For DRL 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 is used as the log subdisk should not contain 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 two or more sectors (preferably an even number, as the last sector in a log subdisk with an odd number of sectors is not 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. 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.

Mirroring and Striping Guidelines

You can follow these general guidelines when using mirroring and striping together. See "Mirroring Plus Striping (RAID-1 + RAID-0)" on page 34 for more information.

- 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 "Striping Guidelines" on page 52.
- Follow the mirroring guidelines described in "Mirroring Guidelines" on page 53.

Striping and Mirroring Guidelines

You can follow these general guidelines when using striping and mirroring together. See "Striping Plus Mirroring (RAID-0 + RAID-1)" on page 34 for more information.

- 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 "Striping Guidelines" on page 52.
- Follow the mirroring guidelines described in "Mirroring Guidelines" on page 53.

RAID-5 Guidelines

You can follow these general guidelines when using RAID-5. See "RAID-5" on page 32 for more information.

In general, the guidelines for mirroring and striping together also apply to RAID-5. The following guidelines should also 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 is 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 is 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.

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. Loss of access can be due to the failure of a key disk (a disk used for system operations). The VERITAS Volume Manager can protect your system from these problems.

To maintain system availability, data important to running and booting your system must be mirrored. The data must be preserved so it can be used in case of failure.

The following are suggestions on protecting your system and data:

- Place the disk containing the root file system (the *root* or *boot* disk) under Volume Manager control through encapsulation. This converts the `root` and `swap` devices to volumes (`rootvol` and `swapvol`). 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 leaves your system unbootable and unusable.

  For maximum availability of the system, create mirrors for the `rootvol`, `swapvol`, `usr`, and `var` volumes. See the Recovery chapter in the *VERITAS Administrator's Reference Guide* 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 still exists on the other disk.

  If you use `vxassist mirror` to create mirrors, it locates the mirrors so 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 file `/etc/default/vxassist` to set the default layout to mirrored.

- Leave the Volume Manager hot-relocation feature enabled to automatically detect failures, notify you of the nature of the failures, attempt to relocate any affected subdisks that are redundant, and initiate recovery procedures. Provide at least one hot-relocation spare disk per disk group so sufficient space is available for relocation in the event of a failure.

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 contain enough contiguous spare or free space for 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 DRL 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. Also, typing a command in error can remove critical files or damage a file system directly.

Volume Manager Operations 3

Introduction

This chapter provides detailed information about VERITAS Volume Manager features.

The following topics are covered in this chapter:

- Online Relayout
- Hot-Relocation
- Volume Resynchronization
- Dirty Region Logging
- Fast Mirror Resynchronization (FMR)
- Volume Manager Rootability
- Dynamic Multipathing (DMP)
- VxSmartSync Recovery Accelerator
- Volume Manager Task Monitor
- Volume Manager Cluster Functionality

Online Relayout

Online Relayout allows you to convert any supported storage layout in the Volume Manager to any other, in place, with *uninterrupted data access*. You usually change the storage layout in the Volume Manager to change the redundancy or performance characteristics of the storage. The Volume Manager adds redundancy to storage either by duplicating the address space (mirroring) or by adding parity (RAID-5). Performance characteristics of storage in the Volume Manager can be changed by changing the striping parameters which are the number of columns and the stripe width.

Layout changes can be classified into these types:

- RAID-5 to mirroring
- mirroring to RAID-5

- adding or removing parity
- adding or removing columns
- changing stripe width

Storage Layout

Online relayout currently supports these storage layouts:

- concatenated
- striped
- RAID-5
- mirrored
- striped-mirror
- concatenated-mirror

Note When using the VERITAS Volume Manager Storage Administrator, `Striped-Pro` is the GUI term used for a striped-mirror, and `Concatenated-Pro` is the GUI term used for a concatenated-mirror.

How Online Relayout Works

The VERITAS online relayout feature allows you to change storage layouts that you have already created in place, without disturbing data access. You can change the performance characteristics of a particular layout to suit changed requirements.You can transform one layout to another by invoking a single command.

A striped-mirror plex is a striped plex on top of a mirrored volume, resulting in a single plex that has both mirroring and striping. This combination forms a plex called a *striped-mirror plex*. A concatenated plex can be mirrored in the same way. Online Relayout supports transformations to and from striped-mirror and concatenated-mirror plexes.

Note Changing the number of mirrors during a transformation is not currently supported.

For example, you may have a striped layout with a 128K stripe unit size that may not be providing optimal performance. You can change the stripe unit size of the layout by using the relayout feature.

File systems mounted on the volumes do not need to be unmounted to achieve this transformation as long as the file system provides online shrink and grow operations. VFS provides these features.

Online relayout reuses the existing storage space and has space allocation policies to address the needs of the new layout. The layout transformation process converts a given volume to the destination layout by using minimal temporary space.

The transformation is done by moving a portion-at-a-time of data in the source layout to the destination layout. Data is copied from the source volume to the temporary space. Data is remove from the source volume storage area in portions. The source volume storage area is then transformed to the new layout and data saved in the temporary space is written back to the new layout. This operation is repeated until all the storage and data in the source volume has been transformed to the new layout.

You can use Online Relayout to change the number of columns, change stripe width, remove and add parity, and change RAID-5 to mirroring.

Types of Transformation

At least one of the following criteria must be met for an onliner relayout operation to be effective. You must perform one or more of the following operations:

- Change RAID-5 to mirroring
- Change RAID-5 from mirroring
- Change the number of columns
- Change the stripe width
- Remove or add parity

To be eligible for layout transformation, mirrored volume plexes must be identical in layout, with the same stripe width and number of columns. See Table 1, ”Supported Layout Transformations.”

Table 1. Supported Layout Transformations

From/To	Striped Mirrored	Concatenated Mirrored	Regular Mirrored	RAID-5	Concatenated	Striped
Striped Mirrored	Yes 1	Yes 2	No 3	Yes 4	Yes 5	Yes 6
Concatenated Mirrored	Yes 7	No 8	No 9	Yes 10	No 11	Yes 12
Regular Mirrored	Yes 13	Yes 14	No 15	Yes 16	No 17	No 18
RAID-5	Yes 4	Yes 10	No 19	Yes 20	Yes 21	Yes 22

Table 1. Supported Layout Transformations

From/To	Striped Mirrored	Concatenated Mirrored	Regular Mirrored	RAID-5	Concatenated	Striped
Concatenated	Yes 5	No 11	No 17	Yes 21	No 23	Yes 24
Striped	Yes 6	Yes 12	No 18	Yes 22	Yes 24	Yes 25

Entries in Table 1 include the following:

- *Yes*—indicates that an Online Relayout operation is possible.
- *No*—indicates that the operation *may* be possible but you cannot use Relayout.
- *Numbers*—indicate a brief description of the possible changes in that particular layout transformation. See the following "Number Descriptions."
- *Operations*—can be performed in both directions.

Number Descriptions

The numbers in Table 1 describe the relayout operation as follows:

1. Changes the stripe width or number of columns.
2. Removes all of the columns.
3. Not a Relayout, but a convert operation.
4. Changes mirroring to RAID-5 and/or stripe width/column changes.
5. Changes mirroring to RAID-5 and/or stripe width/column changes.
6. Changesstripe width/column and remove a mirror.
7. Adds columns.
8. Not a Relayout operation.
9. A convert operation.
10. Changes mirroring to RAID-5. See the `vxconvert` procedure.
11. Removes a mirror; not a Relayout operation.

12. Removes a mirror and add striping.

13. Changes an old mirrored volume to a stripe mirror. Relayout is valid only if there are changes to columns / stripe width; otherwise, this is a convert operation. See the `vxconvert` procedure.

14. Changes an old mirrored volume to a concatenated mirror. Relayout is valid only if there are changes to columns; otherwise, this is a convert operation.

15. No changes; not a Relayout operation.

16. Changes an old mirrored volume to RAID-5. Choose a plex in the old mirrored volume to use Relayout. The other plex is removed at the end of the Relayout operation.

17. Unless you choose a plex in the mirrored volume and change the column / stripe width, this is not a Relayout operation.

18. Unless you choose a plex in the mirrored volume and change the column / stripe width, this is not a Relayout operation.

19. Not a Relayout operation.

20. Changes stripe width / column.

21. Removes parity and all columns.

22. Removes parity.

23. No changes; not a Relayout operation.

24. Removes columns.

25. Changes stripe width / number of columns.

Transformation Characteristics

Transformation of data from one layout to another involves rearrangement of data in the existing layout to the new layout. During the transformation, Online Relayout retains data redundancy by mirroring any temporary space used. Read / write access to data is not interrupted during the transformation.

Data is not corrupted if the system fails during a transformation. The transformation continues after the system is restored and read / write access is maintained.

You can reverse the layout transformation process at any time, but the data may not be returned to the exact previous storage location. Any existing transformation in the volume should be stopped before doing a reversal.

You can determine the transformation direction by using the `vxrelayout status` command.

These transformations eliminate I/O failures as long as there is sufficient redundancy and space to move the data.

Transformations and Volume Length

Some layout transformations can cause the volume length to increase or decrease. If the layout transformation causes the volume length to increase or decrease, Online Relayout uses `vxresize` to shrink or grow a file system.

Sparse plexes are not transformed by online relayout, and no plex can be rendered sparse by online relayout.

Note Online relayout can be used only with volumes created with the `vxassist` command or the Storage Administrator.

The following transformations are not supported:

- Transformation of log plexes
- A snapshot of a volume when there is an Online Relayout operation running on the volume

Hot-Relocation

Hot-relocation allows a system to automatically react to I/O failures on redundant (mirrored or RAID-5) Volume Manager objects and restore redundancy and access to those objects. The Volume Manager detects I/O failures on objects and relocates the affected subdisks. The subdisks are relocated to disks designated as *spare disks* and/or free space within the disk group. The Volume Manager then reconstructs the 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. Existing volumes on 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. Nonredundant subdisks on a failed disk are not relocated, but the system administrator is notified of the failure.

How Hot-Relocation Works

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

The hot-relocation daemon, `vxrelocd`, monitors Volume Manager for events that affect redundancy and performs 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 a failure occurs, it triggers a hot-relocation attempt.

A successful hot-relocation process involves:

1. Detecting Volume Manager events resulting from the failure of a disk, plex, or RAID-5 subdisk.

2. Notifying the system administrator (and other designated users) of the failure and identifying the affected Volume Manager 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 outcome of the recovery attempt.

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

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, Volume Manager 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 free space mentioned in hot-relocation is always the free space not excluded from hot-relocation use. Disks can be excluded from hot-relocation use by using the Storage Administrator interface: `vxdiskadm` or `vxedit`.

The system administrator can designate one or more disks as hot-relocation spares within each disk group. Disks can be designated as spares by using the Storage Administrator interface, `vxdiskadm`, or `vxedit`. 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 Volume Manager 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.

To determine which disk from among the elibilble ones should be used, hot-relocation tries to use the disk that is “closest” to the failed disk. The value of “closeness” depends on the controller, target, and disk number of the failed disk. A disk on the same controller as the failed disk is closer than a disk on a different controller; a disk under the same target as the failed disk is closer than one on a different target.

Hot-relocation tries to move all subdisks from a failing drive to the same destination disk, if possible.

If the failing disk is a root disk, hot-relocation shall only work if it can relocate all of the filesystems on to the same disk. If none are found,the system administrator will be notified via Email.

When hot-relocation takes place, the failed subdisk is removed from the configuration database and Volume Manager ensures that the disk space used by the failed subdisk is not recycled as free space.

Refer to the *VERITAS Volume Manager Installation Guide* for information on how to disable hot-relocation.

Unrelocate Utility

VxVM Hot-relocation allows the system to automatically react to I/O failures on a redundant VxVM object at the subdisk level and take necessary action to make the object available again. This mechanism detects I/O failures in a subdisk, relocates the subdisk, and recovers the plex associated with the subdisk. After the disk has been replaced, Volume Manager provides the `vxunreloc` utility, which can be used to restore the system to the same configuration that existed before the disk failure. `vxunreloc` allows you to move the hot-relocated subdisks back onto a disk that was replaced due to a disk failure.

Volume Resynchronization

When storing data redundantly, using mirrored or RAID-5 volumes, the Volume Manager ensures that all copies of the data match exactly. However, under certain conditions (usually due to complete system failures), some redundant data on a volume can become inconsistent or *unsynchronized*. The mirrored data is not exactly the same as the original data. Except for 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. This can result in the data becoming unsynchronized. 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.

The Volume Manager needs to ensure 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 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 outstanding writes and do not require resynchronization.

The Volume Manager records when a volume is first written to and marks it as *dirty*. When a volume is closed by all processes or stopped cleanly by the administrator, all writes have been 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 process of resynchronization depends on the type of volume. RAID-5 volumes that contain RAID-5 logs can “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 done by placing the volume in recovery mode (also called *read-writeback recovery mode*). Resynchronizing of data in the volume is done in the background. This allows the volume to be available for use while recovery is taking place.

The process of resynchronization can be expensive and can impact system performance. The recovery process reduces some of this impact by spreading the recoveries to avoid stressing a specific disk or controller.

For large volumes or for a large number of volumes, the resynchronization process can take time. These effects can be addressed by using Dirty Region Logging for mirrored volumes, or by ensuring that RAID-5 volumes have valid RAID-5 logs. For volumes used by database applications, the VxSmartSync™ Recovery Accelerator can be used (see “VxSmartSync Recovery Accelerator” on page 78).

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. DRL 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. Restoration is done 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 recovers 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 plex of the volume. 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 that case, the log subdisk risks becoming unavailable if 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. Then the plex can 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.

Some volumes, such as those used for Oracle replay logs, are written sequentially and do not benefit from this lazy cleaning of the DRL bits. For these volumes, *sequential DRL* can be used to further restrict the number of dirty bits and speed up recovery. The number of dirty bits allowed for sequential DRL will be restricted by the tunable `voldrl_max_dirty`. Using sequential DRL on volumes that are written sequentially may severely impact I/O throughput.

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

Fast Mirror Resynchronization (FMR)

The Fast Mirror Resynchronization (FMR) feature performs quick and efficient resynchronization of stale mirrors by increasing the efficiency of the VxVM snapshot mechanism to better support operations such as backup and decision support. Typically, these operations require that the data store volume is quiescent and/or secondary access to the store not affect or impede primary access (i.e., throughput, updates, consistency, etc.).

To achieve these goals, VxVM provides a *snapshot* mechanism that creates an exact copy of a primary volume at a particular instance in time. After a snapshot is taken, it can be accessed independently of the volume from which it was taken. In a shared/clustered VxVM environment, it is possible to eliminate the resource contention and overhead of using the snapshot by accessing it from a different machine.

Fast Mirror Resynchronization overcomes certain drawbacks that exist in earlier version of the snapshot mechanism. They are:

- After a snapshot is taken, the primary volume and snapshot can diverge. They are no longer consistent. As a result, the snapshot must be discarded when it is no longer useful, and a new snapshot must be taken in order to obtain a current copy of the primary data.
- Snapshot setup time can limit the usefulness of the snapshot feature. This is because the period in which a snapshot can be created is directly proportional to the size of the volume. For large, enterprise class volumes, this period can dictate off-line policies in unacceptable ways.

FMR Components

FMR provides two fundamental enhancements to VxVM. The first is to optimize the mirror resynchronization process (*Fast Mirror Resynchronization*), and the second is to extend the snapshot model (*Fast Mirror Reconnect*) to provide a method by which snapshots can be refreshed and re-used rather than discarded.

Fast Mirror Resynchronization Component

FMR requires keeping track of data store updates missed by mirrors that were unavailable at the time the updates were applied. When a mirror returns to service, it must re-apply only the updates missed by that mirror. With FMR, this process requires an amount of restoration far less than the current method whereby the entire data store is copied to the returning mirror.

An unavailable mirror is one that has been *detached* from its volume either automatically (by VxVM, as the result of an error), or directly by an administrator (via a VxVM utility such as `vxplex` or `vxassist`). A *returning mirror* is a mirror that was previously detached and is in the process of being re-attached to its original volume as the result of `vxrecover` or `vxplex att` operation.

FMR will not alter the current mirror failure and repair administrative model. The only visible effect is that typical mirror repair operations conclude more quickly.

The resynchronization enhancement allows the administrator to enable / disable FMR on a per-volume basis, and to check FMR status.

Fast Mirror Reconnect Component

Fast Mirror Reconnect augments the existing snapshot usage model. Without Fast Mirror Resynchronization, an independent copy of a volume is created via the snapshot mechanism. The original volume and the replica volume are completely independent of each other and their data may diverge.

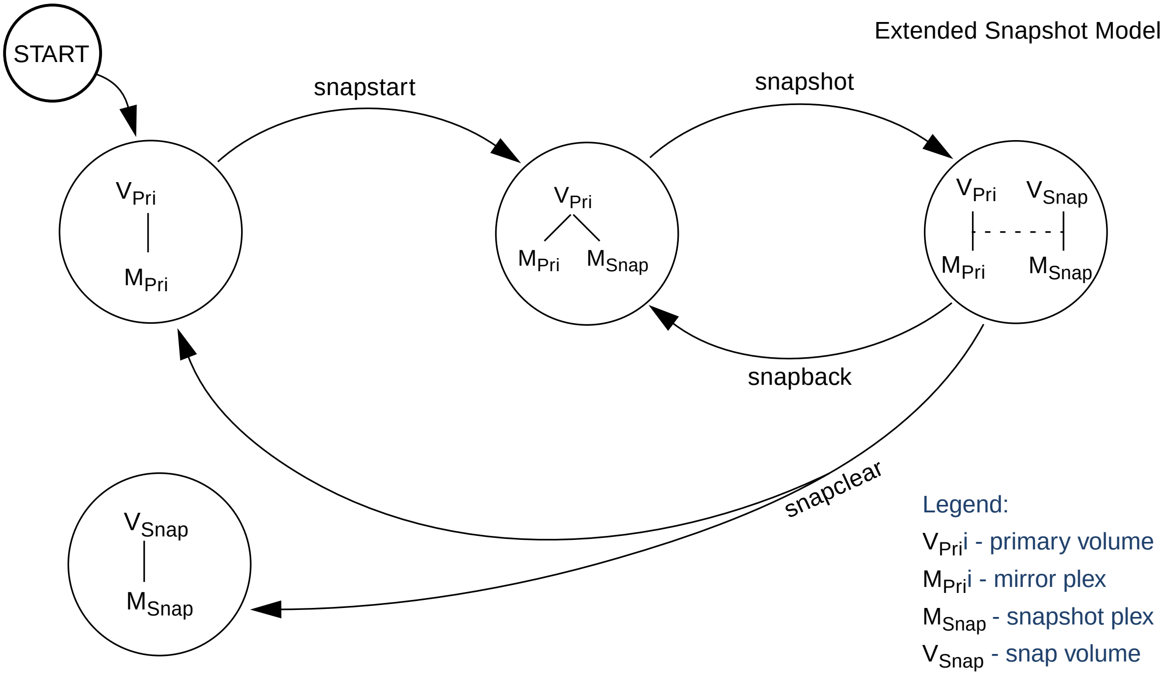
The *Fast Mirror Reconnect* snapshot enhancement makes it possible to re-associate a snapshot volume with its original peer for the express purpose of reducing the work-load required to perform cyclic operations which rely heavily upon the VxVM snapshot functionality.

FMR Enhancements to VxVM Snapshot Functionality

The FMR snapshot enhancements to Release 3.1 extend the snapshot model as shown in Figure 23, ”FMR Enhanced Snapshot,” on page 71. Beginning with Release 3.1, the `snapshot` command behaves as before with the exception that it creates an association between the original volume and the snap volume. A new command, `vxassist snapback`, leverages this association to expediently return the snapshot plex (M_{Snap}) to the volume from which it was snapped (in this example, V_{Pri}).

Figure 23, ”FMR Enhanced Snapshot,” depicts the extended transitions of the snapshot rmodel as introduced by the `snapback` and `snapclear` commands.

Figure 23. FMR Enhanced Snapshot



Additionally, a new command, `vxassist snapclear`, relieves a volume of the administrative overhead of tracking a snapshot by permanently destroying the association created by the `snapshot` command. This capability is useful in situations where it is known that a snapshot will never return to the volume from which it was created.

Theory of Operation

The basis of FMR lies in change tracking. Keeping track of updates missed when a mirror is offline / detached / shapshotted, and then applying only those updates when the mirror returns, considerably reduces the time to resynchronize the volume.

The basis for this change tracking is the use of a bitmap. Each bit in the bitmap represents a contiguous region (an extent) of a volume’s address space. This contiguous region is called the *region size*. Typically, this region size is one block; each block of the volume is represented by one bit in the bitmap. However, a tunable called `vol_fmr_logsz` is provided, which can be used to limit the maximum size (in blocks) of the FMR map. When computing the size of the map, the algorithm starts with a region size of one, and if the resulting map size is less than the `vol_fmr_logsz`, then the computed value becomes the map size. If the size is larger than `vol_fmr_logsz`, an attempt is made to accommodate `vol_fmr_logsz` with a region size of two, and so on until the map size is less than the `vol_fmr_logsz` tunable.

For example:

volume size = 1G

`vol_fmr_logsz` = 4

on a system with a block size of 512 bytes, that is 4*512=2048 bytes or 2048*8 = 16384 bits.

Thus, for a 1G volume, a region size of one is 2048 bits, which is less than four blocks, so the map size is 2048 bits or 256 bytes.

Note that if the size of the volume increases, this computation is redone to ensure the map size does not exceed the `vol_fmr_logsz` tunable.

Persistent Versus Non-Persistent Tracking

For VxVM 3.1, the FMR maps are allocated in memory. They do not reside on disk or persistent store, unlike a DRL. Therefore, if the system crashes, this information is lost and the entire length of the volume will be re-synchonized in the case of any reboot either as a result of a crash or because of a planned stoppage of the operating system. One advantage of this approach is that the FMR updates (updates to this map) do not cost anything in terms of performance, as no disk updates must be done. However, if the system crashes, this information is lost and full resynchronization of mirrors is once again necessary.

Snapshot(s) and FMR

To take advantage of the FMR delta tracking when using snapshots, use the new snapshot option. After a snapshot is taken, the snapshot option is used to reattach the snapshot plex. If FMR is enabled before the snapshot is taken and is not disabled at any time before the snapshot is complete, then the FMR delta changes reflected in the FMR bitmap are used to resynchronize the volume during the snapback. To make it easier to create snapshots of several volumes at the same time, the snapshot option has been enhanced to accept more than one volume and a naming scheme has been added. By default, each replica volume is named `SNAP-`<*original volume*>. This default can be overridden with options on the command line. Only volumes in the same disk group can take a snapshot of more than one volume at a time.

To make it easier to snapshot all the volumes in a single disk group, a new option has been added to `vxassist`, but it fails if any of the volumes in the disk group do not have a complete snapshot plex. It is possible to take several snapshots of the same volume. A new FMR bitmap is produced for each snapshot taken, and the resynchronzation time for each snapshot is minimized.

The snapshot plex can be chosen as the preferred set of data when performing a snapback. Adding `-o resyncfromreplica` to the snapback option copies the data on the snapshot (replica) plex onto all the mirrors attached to the original volume. By default, the data on the original volume is preferred and copied onto the snapshot plex.

It is possible to grow the replica volume, or the original volume, and still use FMR. Growing the volume extends the bitmap FMR uses to track the delta changes. This may change the size of the bitmap or its region size. In either case, the part of the bitmap that corresponds to the grown area of the volume is marked as “dirty” so that this area is resynchronized. The snapshot operation fails if the snapshot attempts to create an incomplete snapshot plex. In such cases, it is necessary to grow the replica volume, or the original volume, before the snapback option is run. Growing the two volumes separately can lead to a snapshot that shares physical disks with another mirror in the volume. To prevent this, grow the volume after the `snapback` command is complete.

Any operation that changes the layout of the replica volume can mark the FMR map for that snapshot “dirty” and require a full resynchronzation during the snapback. Operations that cause this include subdisk split, subdisk move, and online relayout of the replica. It is safe to perform these operations after the snapshot is completed. For more information, see the `vxvol` (1M), `vxassist` (1M), and `vxplex` (1M) manual pages.

FMR and Writeable Snapshots

One of two options is used to track changes to a writeable snapshot, as follows:

- create of a separate map that tracks changes to a snapshot volume
- update the map of the parent of the snapshot volume. Use this shortcut method only if there are few updates to the snapshot volume, such as in the backup and DSS (decision support systems) applications

For VxVM 3.1, the latter method is implemented; i.e., the map of the parent of the snapshot volume is updated when writing to the snapshot.

FMR and Layered Volumes

Enabling FastResync on a layered volume will enable only a parent volume (top of volume). To enable FastResync on subvolumes, it must be enabled on all subvolumes.

Caveats and Limitations

FMR is not supported on RAID-5 volumes.

When a subdisk is relocated, the entire plex is marked “dirty” and a full resynchronzation becomes necessary.

Volume Manager Rootability

The Volume Manager can place various files from different systems (for example, the root file system, `swap` device, `usr` file system, and `stand` 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 Volume Manager control through the process of *encapsulation*.

Encapsulation converts existing partitions on that disk to volumes. Once under Volume Manager control, the `root` and `swap` devices appear as volumes and provide the same characteristics as other Volume Manager 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*; and a volume that contains the stand file system is referred to as a *stand volume*.

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

Mirroring disk drives critical to booting ensures that no single disk failure renders the system unusable. A suggested configuration is to mirror the critical disk onto another available disk (using the `vxdiskadm` command). If the disk containing the `root`, `stand`, 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 Chapter 7, “*Recovery*,”

Booting With Root Volumes

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

To avoid this restriction, the mirrors of the `rootvol`, `standvol`, and `swapvol` volumes are accessed by the system during startup. During startup, the system sees the `rootvol`, `standvol`, and `swapvol` volumes as regular partitions and accesses them using standard partition numbering. `rootvol`, `standvol`, and `swapvol` volumes are 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`, `standvol`, or `swapvol` volume. Any mirrors of these volumes needed for booting cannot be striped or spanned.

Boot-time Volume Restrictions

The `rootvol`, `standvol`, `swapvol`, 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` can 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. Also, `swapvol` has minor device number 1. The `usr` volume does not have a specific minor device number. See *Recovery* in Chapter 7.
- Restricted mirrors of `rootvol`, `var`, `usrrootvol`, `standvol`, and `swapvol` devices have “overlay” partitions created for them. An overlay partition is one that exactly includes the disk space occupied by the restricted mirror. During boot, before the `rootvol`, `var`, `usrrootvol`, `standvol`, and `swapvol` volumes are fully configured, the default volume configuration uses the overlay partition to access the data on the disk.
- 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`, `standvol`, and `swapvol` cannot be spanned or contain a primary plex with multiple noncontiguous subdisks. You cannot grow or shrink any volume associated with an encapsulated bootdisk (`rootvol`, `usr`, `var`, `opt`, `swapvol`, etc.) because these map to a physical underlying partition on the disk and must be contiguous. A workaround is to unencapsulate the bootdisk, repartition the bootdisk as desired (growing or shrinking partitions as needed), and then re-encapsulating.
- 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`, `standvol`, `swapvol`, and `usr` cannot be Dirty Region Logging volumes.

In addition to these requirements, it is a good idea to have at least one contiguous, (cylinder-aligned if appropriate) 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).

Dynamic Multipathing (DMP)

On some systems, the Volume Manager 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 sound connections to the disk. DMP 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. A disk that is part of a disk array is represented by a metadevice that has two physical access paths. You can use the Volume Manager administrative utilities such as `vxdisk` to display all the paths of a metadevice and status information of the various paths.

A multipathing condition can exist when a physical disk can be accessed by more than one operating system device handle. Each multipath operating system device handle permits data access and control through alternate host-to-device pathways.

Volume Manager can be configured with its own DMP system to organize access to multipathed devices. Volume Manager detects multipath systems by using the Universal World-Wide-Device Identifiers (WWD IDs). The physical disk must provide unambiguous identification through its WWD ID for DMP to access the device.

If DMP cannot identify the physical disk through its WWD ID, identification is left to the Volume Manager device detection methods. Device detection depends on Volume Manager recognizing on-disk metadata identifiers.

Volume Manager DMP creates metanodes representing metadevices for each multipath target that it has detected. Each metanode is mapped to a set of operating system device handles and configured with an appropriate multipathing policy. Volume Manager DMP creates metanodes for all attached physical disks accessible through an operating system device handle.

Volume Manager DMP manages multipath targets, such as disk arrays, which define policies for using more than one path. Some disk arrays permit more than one path to be concurrently active (Active / Active). Some disk arrays permit only one path to be active, holding an alternate path as a spare in case of failure on the existing path (Active / Passive). Some disk arrays have more elaborate policies.

In general, Volume Manager is designed so that the VM disk is mapped to one Volume Manager DMP metanode. To simplify VxVM logical operations, each VM disk is mapped to a unique Volume Manager DMP metanode. The mapping occurs whether or not the physical disk device is connected in a multipathing configuration.

Path Failover Mechanism

DMP enhances system reliability when used with multiported disk arrays. In the event of the loss of one connection to the disk array, DMP automatically selects the next I/O paths for the I/O requests dynamically without action from the administrator.

DMP allows the administrator to indicate to the DMP subsystem in the Volume Manager whether the connection is repaired or restored. This is called DMP reconfiguration.The reconfiguration procedure also allows the detection of newly added devices, as well as devices that are removed after the system is fully booted (only if the operating system sees them properly).

Load Balancing

To provide load balancing across paths, DMP follows the *balanced path mechanism* for active/active disk arrays. Load balancing makes sure that I/O throughput can be increased by using the maximum bandwidth of all paths to the maximum. However, sequential I/Os to a disk are sent down the same path to optimize I/O throughput. This is done in order to use the effect of disk track caches.

For active/passive disk arrays, I/Os are sent down the primary path until it fails. Once the primary path fails, I/Os are then switched over to the other available primary paths or secondary paths. To avoid the continuous transfer of ownership of LUNs from one controller to another, which results in severe I/O slowdown, load balancing across paths is not done for active/passive disk arrays .

Booting From DMP Devices

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

Enabling and Disabling Controllers

DMP allows the administrator to turn off I/Os to a host I/O controller to perform administrative operations. It can be used for maintenance of controllers attached to the host or a disk array supported by Volume Manager. I/O operations to the host I/O controller can be turned on after the maintenance task is completed. This operation can be accomplished using the `vxdmpadm` command provided with Volume Manager.

For example, if the system has a StorEdge A5000(TM) array and the user needs to change an A5000 Interface Board connected to this disk array, the `vxdmpadm` command should be used to get a list of host I/O controllers connected on this A5000 interface board. These should be disabled. Once this is done, further I/Os to the disks which are accessed through these controller(s) is stopped.

The Interface Board can then be replaced without causing any disruption to ongoing I/Os to disks present on this disk array. This is required because normally, for active/active type disk arrays (as in this example), Volume Manager uses the balanced path mechanism to schedule I/Os to a disk with multiple paths to it. As a result, I/Os may go through any path at any given point in time.

For active/passive type disk arrays, I/Os will be scheduled by Volume Manager to the primary path until a failure is encountered. Therefore, to change an interface card on the disk array or a card on the host (when possible) that is connected to the disk array, the I/O operations to the host I/O controller(s) should be disabled. This allows all I/Os to be shifted over to an active secondary path or an active primary path on another I/O controller before the hardware is changed.

After the operation is over, the paths through these controller(s) can be put back into action by using the enabled option of the `vxdmpadm` command.

Volume Manager does not allow you to disable the last active path to the root disk.

Displaying DMP Database Information

The `vxdmpadm` command can be used to list DMP database information and perform other administrative tasks. This command allows you to list all the controllers on the systems (connected to disks) and other related information stored in the DMP database. This information can be used to locate system hardware and make a decision regarding which controllers to enable/disable.

`vxdmpadm` also provides you with other useful information such as disk array serial number and the list of DMP devices (disks) that are connected to the disk array, the list of paths that go through a particular controller, etc.

VxSmartSync Recovery Accelerator

The VxSmartSync™ Recovery Accelerator is available for some systems. VxSmartSync for Mirrored Oracle Databases is a collection of features that speed up the resynchronization process (known as *resilvering*) for volumes used in with the Oracle Universal Database. These features use an extended interface between Volume Manager volumes and the database software so they can avoid unnecessary work during mirror resynchronization. These extensions can result in an order of magnitude improvement in volume recovery times.

Oracle automatically takes advantage of SmartSync when it is available.

The system administrator must configure the volumes correctly to use VxSmartSync. For Volume Manager, there are two types of volumes used by the database:

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

VxSmartSync works with 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.

Data Volume Configuration

The improvement in recovery time for data volumes is achieved by letting the database software decide which portions of the volume require recovery. The database keeps logs of changes to the data in the database and can determine which portions of the volume require recovery. By reducing the amount of space that requires recovery and allowing the database to control the recovery process, the overall recovery time is reduced.

Also, 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 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 Volume Manager to do 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.

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, Volume Manager can 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 reduces recovery time impact on the system.

The enhanced interfaces for redo logs allow the database software to inform Volume Manager when a volume is to be used as a redo log. This allows Volume Manager to modify the DRL behavior of the volume 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.

Volume Manager Task Monitor

The Volume Manager Task Monitor tracks the progress of system recovery by monitoring task creation, maintenance, and completion. The Task Monitor allows you to monitor task progress and to modify characteristics of tasks, such as pausing and recovery rate (for example, to reduce the impact on system performance). You can also monitor and modify the progress of the online relayout feature. See "Online Relayout," for more information.

Volume Manager Cluster Functionality

The Volume Manager includes an *optional* cluster feature that enables VxVM to be used in a cluster environment. The cluster functionality in the Volume Manager allows multiple hosts to simultaneously access and manage a given set of disks under Volume Manager control (*VM disks*).

A *cluster* is a set of hosts sharing a set of disks; each host is referred to as a *node* in the cluster. The nodes are connected across a network. If one node fails, the other node(s) can still access the disks. The Volume Manager cluster feature presents the same logical view of the disk configurations (including changes) on all nodes. When the cluster feature is enabled, Volume Manager objects are capable of being shared by all of the nodes in a cluster.

For more information about the cluster functionality in the Volume Manager, refer to the chapter on cluster functionality in this manual.

Note The cluster functionality in Volume Manager is a separately licensable feature.

Disk Tasks 4

Introduction

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

Note Most Volume Manager commands require superuser or other appropriate privileges.

The following topics are covered in this chapter:

- Standard Disk Devices
- Disk Groups
- Disk and Disk Group Commands
- Initializing and Adding Disks
- Adding a Disk to Volume Manager
- Adding a Disk to a Disk Group
- Removing a Disk from a Disk Group
- Moving Disks
- Renaming a Disk
- Reserving Disks
- Taking a Disk Offline
- Mirroring a Disk
- Removing a Disk
- Displaying Disk Information
- Detecting and Replacing Failed Disks
- Creating a Disk Group
- Upgrading a Disk Group

- Removing a Disk Group
- Moving Disk Groups Between Systems
- Destroying a Disk Group
- Using Special Devices
- vxdiskadm Menu Interface Tasks

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. These partitions are named, in order, 0 through 7. Partition 2 is reserved to indicate the entire disk.

Note On some systems, Volume Manager supports up to sixteen partitions. On these systems, the partitions are named 0 through 15 and partition 0 is reserved to indicate the entire disk.

When a partition is placed under Volume Manager control, a VM disk is assigned to that partition. You can use a symbolic name (the *disk name* or *disk media name*), for example, `disk0`, to refer to a VM disk.

Note Your system may use a *device name* that differs from the examples.

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

- `c#` —The number of the controller to which the disk drive is attached
- `b#`—The corresponding bus (if used on your system)
- `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` refers to the standard partitioning method used by Volume Manager. On some systems, Volume Manager uses s0 as the standard partitioning method. The physical disk is identified to the Volume Manager as `c#b#t#d#s#` (`b#` is for systems that use a bus).

For many commands, the suffix s# is optional, though display commands report devices with an s# suffix. The Volume Manager vxdiskadm and vxdiskadd utilities take device names without the s2 (or s0) 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 has two regions:

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

Three basic disk types are used by Volume Manager:

- *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 (or s0), the default type is sliced.

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 has 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 use more than one disk group, unless they have a large number of disks. Disks are not added to disk groups until the disks are needed to create Volume Manager objects. Disks can be initialized, reserved, and added to disk groups later. However, at least one disk (partition) must be added to rootdg for you to do the Volume Manager installation procedures.

When a disk is added to a disk group, it is given a name (for example, disk02). This name identifies a disk for volume operations: 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 continues 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 large disk groups can 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 private region is space for a disk group configuration database containing records for each Volume Manager object in that disk group. Because each configuration record takes up 256 bytes (or half a block), the number of records that can be created in a disk group is twice the configuration database copy size. The copy size can be obtained from the output of the command `vxdg list` *`diskgroupname`*.

Disk and Disk Group Commands

The Volume Manager provides several interfaces that you can use to manage disks:

- the graphical user interface
- a set of command-line commands
- the `vxdiskadm` menu-based interface
- the unrelocate command

Commands discussed in this chapter include:

- `vxdiskadm`—this is the Volume Manager Support Operations menu interface. This command provides a menu of disk operations. Each entry in the main menu leads you through a particular task by providing you with information and prompts. Default answers are provided for many questions so you can easily select common answers. See the `vxdiskadm`(1M) manual page for information on how to use `vxdiskadm`.
- `vxdiskadd`—this command 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 prompts. See the `vxdiskadd`(1M) manual page for information on how to use `vxdiskadd`.
- `vxdisk`—this command administers disks under VxVM control. `vxdisk` defines special disk devices, initializes information stored on disks (that the Volume Manager uses to identify and manage disks), and performs additional special operations. See the `vxdisk`(1M) manual page for information on how to use `vxdisk`.
- `vxdg`—this command operates on disk groups. `vxdg` creates new disk groups, and administers existing disk groups. See the `vxdg`(1M) manual page for information on how to use `vxdg`.
- `vxunrelocate`—this command moves subdisks that have been relocated by the hot-relocation feature back to their original disks.

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

Most Volume Manager commands allow you to specify a disk group using the –g option. For example, to create a volume in disk group mktdg, use the following command:

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

The (block) volume device for this volume is:

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

The disk group does not have to be specified if the object names are unique. Most Volume Manager commands use object names specified on the command line to determine the disk group for the operation. For example, to create a volume on disk mktdg01 without specifying the disk group name, use the following command:

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

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

Initializing and Adding Disks

Disks are either initialized or encapsulated when added to VxVM. Encapsulation preserves exisitng data on disks, but initialization destroys existing data.

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.

Formatting the Disk Media

To perform the first initialization phase, use the interactive format (on some systems, diskadd) command to do a media format of any disk.

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

Volume Manager Disk Installation

You use either the `vxdiskadm` menus or `vxdiskadd` to do the disk initialization phase. 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 , “Menu Interface Operations,“ .

You can use `vxdiskadd` to initialize a specific disk. For example, to initialize the second disk on the first controller, use the following command:

```
# vxdiskadd c0t1d2
```

`vxdiskadd` examines your disk to determine whether it has been initialized and displays prompts based on what it finds. `vxdiskadd` checks for disks that can be encapsulated (see “Using vxdisk for Special Encapsulations” on page 120), disks that have already been added to the Volume Manager, and for other conditions.

Note If you are adding an uninitialized disk, warning and error messages are 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, enter **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 are prompted with this display:

```
You can choose to add this disk to an existing disk group, a
new disk group, or leave the disk available for use by futur
add or replacement operations.  To create a new disk group
select a disk group name that does not yet exist.  To leav
the disk available for future use, specify a disk group nam
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), enter **none**. After this, you are prompted 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, enter **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
```

At the following prompt, enter **n** to indicate that this disk should not be excluded from hot-relocation:

```
Exclude another disk from hot-relocation use? [y,n,q,?]
[default:n] n
```

Enter **y** or press Return to continue with the operation after this display:

```
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 saved, enter **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` prompts you to initialize the disk instead, enter **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. You may also be given the option of performing surface analysis on some systems.

```
Initializing device c0t1d0.

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

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

Adding a Disk to Volume Manager

You must place a disk under Volume Manager control and add it to a disk group, before you can use the disk space for volumes. If the disk was previously in use, but not under Volume Manager control, you can preserve existing data on the disk while still letting the Volume Manager take control of the disk. This can be done by using the encapsulation feature of the Volume Manager. Encapsulation preserves any data on the disk. If the disk is new, it must be initialized. Initialization destroys any existing data on the disk. If the disk was previously not under Volume Manager control, but no data is required to be preserved, it should be initialized.

To add a disk, use the following command:

```
# vxdiskadd devname
```

where *devname* is the device name of the disk to be added.

To add the device `c1t0d0` to Volume Manager control, do the following:

1. Enter the following to start `vxdiskadd`:

   ```
   # vxdiskadd c1t0d0
   ```

2. To continue with the task, enter **y** (or press Return) at the following prompt:

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

   Here is the disk selected.  Output format: [Device_Name]
   c1t0d0

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

3. At the following prompt, specify the disk group to which the disk should be added or press Return to accept `rootdg`: (See “Taking a Disk Offline” on page 95 for more information.)

   ```
   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)
   ```

4. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

5. When prompted as to whether this disk should become a hot-relocation spare, enter **n** (or press Return):)

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

   (See “vxdiskadm Menu Interface Tasks” on page 122” for more information.

6. At the following prompt, enter **n** to indicate that this disk should not be excluded from hot-relocation:

```
Exclude another disk from hot-relocation use? [y,n,q,?]
[default:n] n
```

7. To continue with the task, enter **y** (or press Return) at the following prompt:

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

c1t0d0

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

8. If there is data on this disk that needs to be preserved, enter **y** to select encapsulation:

```
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]

c1t0d0

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

9. To continue the task, press Return at the following prompt:

```
The following disk has been selected for encapsulation.
Output format: [Device_Name  c1t0d0]

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

   A message similar to the following indicates that the disk is being encapsulated for Volume Manager use:

```
The disk device c1t0d0 will be encapsulated and added to the
disk group rootdg with the disk name disk01.
```

```
The c1t0d0 disk has been configured for encapsulation.

The first stage of encapsulation has completed successfully.
You should now reboot your system at the earliest possible
opportunity.
The encapsulation will require two or three reboots which will
happen automatically after the next reboot.  To reboot execute
the command:

shutdown -g0 -y -i6

This will update the /etc/vfstab file so that volume devices
are used to mount the file systems on this disk device.  You
will need to update any other references such as backup scripts,
databases, or manually created swap devices.

Goodbye.
```

Remember to perform a shutdown and reboot as soon as convenient.

Adding a Disk to a Disk Group

You can add a new disk to an already established disk group. For example, the current disks have insufficient space for the application or work group requirements, especially if these requirements have changed.

To add an initialized disk to a disk group, use the following command:

```
# vxdiskadd devname
```

To add device `c1t1d0` to `rootdg`, use the following steps:

1. Enter this command to start `vxdiskadd`:

   ```
   # vxdiskadd c1t1d0
   ```

   `vxdiskadd` displays the following message:

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

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

   c1t1d0

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

2. At the following prompt, specify the disk group to which the disk should be added or press Return to accept `rootdg`:

```
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)
```

3. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

4. When `vxdiskadd` asks whether this disk should become a hot-relocation spare, enter **n** (or press Return):

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

5. At the following prompt, enter **n** to indicate that this disk should not be excluded from hot-relocation:

   ```
   Exclude another disk from hot-relocation use? [y,n,q,?]
   [default:n] n
   ```

6. To continue with the task, enter **y** (or press Return) at the following prompt:

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

   ```
   c1t1d0
   ```

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

7. The following prompt indicates that this disk has been previously initialized for future Volume Manager use; enter **y** to confirm that you now want to use this disk:

   ```
   The following disk device appears to have been initialized
   already. The disk is currently available as a replacement disk.
   Output format: [Device_Name]
   ```

   ```
   c1t1d0
   ```

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

8. To reinitialize the disk, enter **y** (or press Return) at the following prompt:

   ```
   The following disk you selected for use appears to already have
   been initialized for the Volume Manager.  If you are certain
   the disk has already been initialized for the Volume Manager,
   then you do not need to reinitialize the disk device.
   ```

```
Output format: [Device_Name]

c1t1d0

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

Messages similar to the following now confirm that this disk is being reinitialized for Volume Manager use. You may also be given the option of performing surface analysis on some systems.

```
Initializing device c1t1d0.

Perform surface analysis (highly recommended)
[y,n,q,?] (default: y) n
Adding disk device c1t1d0 to disk group rootdg with disk name
 disk03.
```

To confirm that the disk has been added to the disk group, enter:

```
# vxdisk list
```

The Volume Manager returns a listing similar to the following:

```
DEVICE     TYPE    DISK    GROUP    STATUS

c0t0d0s2   sliced  disk04  rootdg   online

c1t0d0s2   sliced  disk01  rootdg   online

c1t1d0s2   sliced  disk03  rootdg   online
```

Removing a Disk from a Disk Group

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

```
# vxdg [-g groupname] rmdisk diskname
```

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

For example, to remove `disk02` from `rootdg`, use this command:

```
# 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 allows you to remove the disk even if subdisks are present. For more information, see the `vxdg`(1M) manual page.

Note Use of the `-k` option to `vxdg` can result in data loss.

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

```
# vxdisk rm devicename
```

For example, to remove `c1t0d0` (or `c1b0t0d0`) from Volume Manager control, use these commands:

```
# vxdisk rm c1t0d0s2
```

or on systems with a bus:

```
# vxdisk rm c1b0t0d0s0
```

You can remove a disk on which some subdisks are defined. For example, you can consolidate all 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 this typical message 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 are 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.

Moving Disks

To move 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 use these commands:

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

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

You can also move a disk by using vxdiskadm. Select item 3 (Remove a disk) from the main menu, and then select item 1 (Add or initialize a disk).

Renaming a Disk

If you do not specify a Volume Manager name for a disk, the Volume Manager gives the disk a default name when you add the disk to Volume Manager control. The Volume Manager name is used by the Volume Manager to identify the location of the disk or the disk type. You can change the disk name to reflect a change of ownership or use, enter:

```
# vxedit rename old_diskname new_diskname
```

To rename disk01 to disk03, enter:

```
# vxedit rename disk01 disk03
```

To confirm that the name change took place, enter:

```
# vxdisk list
```

The Volume Manager returns the following:

```
DEVICE     TYPE    DISK    GROUP    STATUS

c0t0d0s2   sliced  disk04  rootdg   online

c1t0d0s2   sliced  disk03  rootdg   online

c1t1d0s2   sliced  -       -        online
```

Note By default, Volume Manager names subdisk objects after the VM disk on which they are located. Renaming a VM disk does not automatically rename the subdisks on that disk.

Reserving Disks

By default, `vxassist` allocates space from any disk that has free space. You can reserve a set of disks for special purposes, such as to avoid general use of a particularly slow or a particularly fast disk.

Use this command to reserve a disk for special purposes:

```
# vxedit set reserve=on diskname
```

After you enter this command, `vxassist` does not allocate space from the selected disk unless that disk is specifically mentioned on the `vxassist` command line. For example, if disk `disk03` is reserved, the command:

```
# vxassist make vol03 20m disk03
```

overrides the reservation and creates a 20 megabyte volume on `disk03`. However, the command:

```
# vxassist make vol04 20m
```

does not use `disk03`, even if there is no free space on any other disk.

To turn off reservation of a disk, enter:

```
# vxedit set reserve=no diskname
```

Taking a Disk Offline

You may need to take a physical disk offline. If the disk is corrupted, you need to disable it and remove it. You also must disable a disk before moving the physical disk device to another location to be connected to another system.

To take a physical disk offline, first remove the disk from its disk group; for more information, see *"Removing a Disk" on page 97*. Then place the disk in an "offline" state using the following command:

```
# vxdisk offline devname
```

Note The device name is used here because the disk is no longer in a disk group and so does not have an administrative name.

Mirroring a Disk

Mirroring the root disk mirrors the root volume and other areas needed for booting onto another disk.

To mirror your boot (root) disk onto another disk, use either the steps outlined here or `vxdiskadm`. This makes it possible to recover from failure of your boot disk by replacing it with the mirror of the boot disk.

Before mirroring your boot disk, the `EEPROM` variable `use-nvramrc?` must be set to `true` if you want to take advantage of the Volume Manager boot disk aliases to identify the mirror of the boot disk if a replacement is needed. If this variable is set to `false`, you will need to determine which disks are bootable yourself. You can set this variable to **`true`** as follows:

```
eeprom use-nvramrc?=true
```

To mirror your boot disk, do the following:

1. Select a disk that is at least as large as your boot disk.

2. Use the `vxdiskadd` command to add the selected disk as a new disk (if it is not already added).

3. Execute the following command:

   ```
   # /etc/vx/bin/vxrootmir alternate_disk
   ```

   where *alternate_disk* is the disk name assigned to the other disk.

   `vxrootmir` creates a mirror for `rootvol` (the volume for the root file system on an alternate disk).

   The alternate boot disk is configured to enable booting from it if the primary boot disk fails.

There may be other volumes on the boot disk, such as volumes for `/home` or `/tmp` file systems. These can be mirrored separately using the `vxassist` utility. For example, if you have a `/home` file system on a volume `homevol`, you can mirror it to *alternate_disk* using the command:

```
# vxassist mirror homevol alternate_disk
```

If you do not have space for a copy of some of these file systems on your alternate boot disk, you can mirror them to other disks. You can also span or stripe these other volumes across other disks attached to your system.

To list all volumes on your primary boot disk, use the command:

```
# vxprint -t -v -e\'aslist.aslist.sd_disk="boot_disk"'
```

To mirror all of the concatenated volumes on this disk to your alternate boot disk, use the command:

```
# /etc/vx/bin/vxmirror boot_disk alternate_disk
```

Removing a Disk

You can remove a disk from a system and move it to another system if the disk is failing or has failed. Before removing the disk from the current system, you must:

1. Unmount any file systems on the volumes.

2. Stop the volumes on the disk.

3. Move the volumes to other disks or back up the volumes. To move a volume, mirror the volume on one or more other disks, then remove the original copy of the volume. If the volumes are no longer needed, they can be removed instead of moved.

To remove a disk, use these steps:

4. Remove the disk from its disk group with this command:

   ```
   # vxdg [-g groupname] rmdisk diskname
   ```

   where *groupname* is the name of the group to which the disk belongs and *diskname* is the name of the disk to be removed.

   For example, to remove `disk01` from `rootdg`, enter:

   ```
   # vxdg rmdisk disk01
   ```

   Since `rootdg` is the default disk group, you do not need to specify it.

5. Remove the disk from Volume Manager control with this command:

   ```
   # vxdisk rm devicename
   ```

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

   ```
   # vxdisk rm c1t0d0s2
   ```

Displaying Disk Information

Before you use a disk, you need to know if it has been initialized and placed under Volume Manager control. You also need to know if the disk is part of a disk group, because you cannot create volumes on a disk that is not part of a disk group. The `vxdisk list` command displays device names for all recognized disks, the disk names, the disk group names associated with each disk, and the status of each disk.

You can display information on all disks that are defined to the Volume Manager with this command:

```
# vxdisk list
```

The Volume Manager returns the following display:

```
DEVICE     TYPE    DISK    GROUP    STATUS

c0t0d0s2  sliced  disk04  rootdg   online

c1t0d0s2  sliced  disk01  rootdg   online

c1t1d0s2  sliced  -       -        online
```

To display details on a particular disk defined to the Volume Manager, enter:

```
# vxdisk list disk01
```

The `vxdisk` utility is used to display the dynamic multipathing information for a particular metadevice. The metadevice is a device representation of a particular physical disk having multiple physical paths from the I/O controller of the system. In the Volume Manager, all the physical disks in the system are represented as metadevices with one or more physical paths.

You can see the multipathing information for a particular metadevice with the command:

```
# vxdisk list device
```

device is the metadevice formed by the DMP subsystem of Volume Manager.

The output shows two paths to a physical device represented by the metadevice `c2t0d0s2`. The path `c2t0d0s2` is active (`state=enabled`) and the other path `c1t0d0s2` is in a failed state (`state=disabled`).

The Volume Manager returns the following display:

```
Device     device
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        state=enabled       type=primary
c1t0d0s2        state=disabled      type=secondary
```

Additional information in the form of *type* is shown for disks on active/passive type disk arrays. This information indicates the *primary* and *secondary* paths to the disk. For example, DG Clariion, Hitachi DF350, etc.

This type information is not present for disks on active/active type disk arrays because there is no concept of primary and secondary paths to disks on these disk arrays. For example, StorEdge A5000 and Sparc Storage Array (SSA) disk arrays.

Detecting and Replacing Failed Disks

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

Hot-Relocation

Hot-relocation automatically reacts to I/O failures on redundant (mirrored or RAID-5) Volume Manager objects and restores redundancy and access to those objects. The Volume Manager detects I/O failures on objects and relocates the affected subdisks to disks designated as spare disks and/or free space within the disk group. Volume Manager then reconstructs the objects that existed before the failure and makes them redundant and accessible again. Refer to “Volume Manager Operations,” for a 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 Volume Manager object. Volume Manager attempts to correct the error. If the error cannot be corrected, Volume Manager tries to access configuration information in the private region of the disk. If it cannot access the 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 by electronic mail of the failure and which Volume Manager objects are affected. `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 except for disks marked `nohotuse` (excluded for hot-relocation use). Once the subdisks are relocated, each relocated subdisk is reattached to its plex.

Finally, `vxrelocd` initiates appropriate recovery procedures. For example, recovery includes 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 enough space is not available (from spare disks or free space and not excluded from hot-relocation with the `nohotuse` flag) 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 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 log plex or a mirrored volume DRL log plex fails, a new log plex is created elsewhere (so the log plex is not actually relocated).

You can 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 “Volume Manager Operations”. If no spares are available at the time of a failure or if there is not enough space on the spares, free space is automatically used if it is not excluded from hot-relocation with the `nohotuse` flag.

By designating spare disks, you have control over which space is used for relocation 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.

By designating a nohotuse disk, you control which free disks are not allowed to be used as replacement disks.

There are two ways to exclude a disk from hot-releocation use. For example:

```
# vxedit -g rootdg set nohotuse=on disk01
# vxdiskadm
```

From the vxdiskadm main menu, select option 15 (Exclude a disk from hot-relocation use)

There are two ways to make a disk available for hot-relocation use. For exam-ple:

```
# vxedit -g rootdg set nohotuse=off disk01
# vxdiskadm
```

From the vxdiskadm main menu, select option 16 (Make a disk available for hot-relocation use)

There are two ways to find out which disks are spares or are excluded from hot-relocation. Examples are:

```
# vxdisl list
# vxprint
```

After a successful relocation occurs, you need to remove and replace the failed disk (see "Replacing Disks"). Depending on the locations of the relocated subdisks, you can choose to move the relocated subdisks elsewhere after hot-relocation occurs (see "Moving Relocated Subdisks" on page 103).

Modifying vxrelocd

Hot-relocation is turned on as long as vxrelocd is running. You 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 (you do not want the free space on some of your disks used for relocation), prevent vxrelocd from starting at system startup time.

Refer to the *VERITAS Volume Manager Installation Guide* for information on how to disable hot-relocation at system startup. 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 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 as follows:

- 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 shown here:

  ```
  # 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 shown here:

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

  where the optional *IOdelay* indicates the desired delay (in milliseconds). The default value for the delay is 250 milliseconds. See the vxrelocd(1M) manual page for more information.

  -O This option is used to revert to an older version. Specifying VxVM_version -O tells vxrelocd to use the relocation scheme in that version.

  -s Before vxrelocd attempts relocation, a snapshot of the current configuration is saved in /etc/vx/saveconfig.d. This option specifies the maximum number of configurations to keep for each disk group. The default is 32.

Displaying Spare Disk Information

You use the command vxdg spare to display information about all of the spare disks available for relocation. The output displays this information:

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

In this example, 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.

Moving Relocated Subdisks

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 can move the relocated subdisks (after hot-relocation is complete) to improve performance.

You can also 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 recreate the configuration that existed before hot-relocation occurred.

During hot-relocation, one of the electronic mail messages sent to `root` is shown in the following example:

```
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 has information about the subdisk before relocation and can be used to decide where to move the subdisk after relocation.

Here is an example message that shows 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 you move any relocated subdisks, fix or replace the disk that failed (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` with this command:

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

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

Using vxunrelocate

VxVM hot-relocation allows the system to automatically react to I/O failures on a redundant VxVM object at the subdisk level and then take necessary action to make the object available again. This mechanism detects I/O failures in a subdisk, relocates the subdisk, and recovers the plex associated with the subdisk. After the disk has been replaced, vxunreloc allows you to restore the system back to the configuration that existed before the disk failure. vxunreloc allows you to move the hot-relocated subdisks back onto a disk that was replaced due to a failure.

When vxunreloc is invoked, you must specify the disk media name where the hot-relocated subdisks originally resided. When vxunreloc moves the subdisks, it moves them to the original offsets. If you try to unrelocate to a disk that is smaller than the original disk that failed,vxunreloc does nothing except return an error.

vxunreloc provides an option to move the subdisks to a different disk from where they were originally relocated. It also provides an option to unrelocate subdisks to a different offset as long as the destination disk is large enough to accommodate all the subdisks.

If vxunreloc cannot replace the subdisks back to the same original offsets, a force option is available that allows you to move the subdisks to a specified disk without using the original offsets. Refer to the vxunreloc(1M) manual page for more information.

The following examples demonstrate the use of vxunreloc.

Example 1:

Assume that disk01 failed and all the subdisks were relocated. After disk01 is replaced, vxunreloc can be used to move all the hot-relocated subdisks back to disk01.

```
vxunreloc -g newdg disk01
```

Example 2:

The vxunreloc utility provides the -n option to move the subdisks to a different disk from where thay were originally relocated.

Assume that disk01 failed, and that all of the subdisks that resided on it were hot-relocated to other disks. vxunreloc provides an option to move the subdisks to a different disk from where they were originally relocated. After the disk is repaired, it is added back to the disk group using a different name, e.g, disk05. If you want to move all the hot-relocated subdisks back to the new disk, the following command can be used:

```
vxunreloc -g newdg -n disk05 disk01
```

Example 3:

Assume that disk01 failed and the subdisks were relocated and that you want to move the hot-relocated subdisks to disk05 where some subdisks already reside. You can use the force option to move the hot-relocated subdisks to disk05, but not to the exact offsets:

```
vxunreloc -g newdg -f -n disk05 disk01
```

Example 4:

If a subdisk was hot relocated more than once due to multiple disk failures, it can still be unrelocated back to its original location. For instance, if disk01 failed and a subdisk named disk01-01 was moved to disk02, and then disk02 experienced disk failure, all of the subdisks residing on it, including the one which was hot-relocated to it, will be moved again. When disk02 was replaced, a vxunreloc operation for disk02 will do nothing to the hot-relocated subdisk disk01-01. However, a replacement of disk01 followed by a vxunreloc operation, moves disk01-01 back to disk01 if vxunreloc is run immediately after the replacement.

After the disk that experienced the failure is fixed or replaced, `vxunreloc` can be used to move all the hot-relocated subdisks back to the disk. When a subdisk is hot-relocated, its original disk media name and the offset into the disk, are saved in the configuration database. When a subdisk is moved back to the original disk or to a new disk using `vxunreloc`, the information is erased. The original `dm` name and the original offset are saved in the subdisk records. To print all of the subdisks that were hot-relocated from `disk01` in the `rootdg` disk group, use the following command:

```
# vxprint -g rootdg -se 'sd_orig_dmname="disk01"'
```

To move all the subdisks that were hot-relocated from `disk01` back to the original disk, type:

```
# vxunreloc -g rootdg disk01
```

The `vxunreloc` utility provides `-n` option to move the subdisks to a different disk from where they were originally relocated. For example, when `disk01` failed, all the subdisks that resided on it were hot-relocated to other disks. After the disk is repaired, it is added back to the disk group using a different name, for example, `disk05`. If you want to move all the hot-relocated subdisks to the new disk, the following command can be used:

```
# vxunreloc -g rootdg -n disk05 disk01
```

The destination disk should have at least as much storage capacity as was in use on the original disk. If there is not enough space, the unrelocate operation will fail and none of the subdisks will be moved.

When `vxunreloc` moves the hot-relocated subdisks, it moves them to the original offsets. However, if there some subdisks existed which occupied part or all of the area on the destination disk, `vxunreloc` will fail. Basically, you have two choices: (1) move the existing subdisks somewhere else, and then re-run `vxunreloc`, or (2) use the `-f` option provided by `vxunreloc` to move the subdisks to the destination disk, but leave it to `vxunreloc` to find the space on the disk. As long as the destination disk is large enough so that the region of the disk for storing subdisks can accommodate all subdisks, all the hot-relocated subdisks will be “unrelocated” without using the original offsets.

Restart vxunreloc After Errors

Internally, vxunreloc moves the subdisks in three phases.The first phase is creates as many subdisks on the specified destination disk as there are the number of the subdisks to be unrelocated. When the subdisks are made, vxunreloc fills in the comment field in the subdisk record with the string “UNRELOC”as an identification. The second phase is the actual data moving. If all the subdisk moves are successful, the third phase proceeds to clean up the comment field of the subdisk records.

Making the subdisk is a all-or-none operation. If vxunreloc cannot make all the subdisks successfully, no subdisk is made and vxunreloc just exits. The operation of the subdisks move iis not all-or-none. One subdisk move is independent of another, and as a result, if one subdisk move fails, the vxunreloc utility prints an error message and then exits. But, all of the subsequent subdisks remain on the disk where they were hot-relocated and will not be moved back. For subdisks that made their way back home, the comment field in their subdisk records is still marked as “UNRELOC” because the cleanup phase is never executed.

If the system goes down after the new subdisks are made on the destination, but before they are moved back, the unrelocate utility can be executed again after the system comes back. As described above, when a new subdisk is created, vxunreloc sets the comment field of the subdisk as “UNRELOC”. When vxunreloc is re-executed, it checks the offset, the len, and the comment fields of the existing subdisks on the destination disk to determine if it was left on the disk at a previous execution of the vxunreloc which will then use it as it sees fit.

Do not manually modify the string “UNRELOC” in the comment field.

If one out of a series of subdisk moves fails, vxunreloc exits. Under this circumstance, you should check the error that caused the subdisk move to fail and determine if the unrelocation can proceed. When you re-execute vxunreloc to resume the subdisk moves, it uses the subdisks created at a previous run.

The cleanup phase is done with one transaction. vxunreloc resets the comment field to a NULL string for all the subdisks marked as “UNRELOC” that reside on the destination disk. This includes clean up for those subdisks that were unrelocated in any previous invocation of vxunreloc, that was not successfully completed.

Detecting Failed Disks

Note The Volume Manager hot-relocation feature automatically detects disk failures and notifies the system administrator of the failures by electronic mail.
If hot-relocation is disabled or you miss the electronic mail, you can see disk failures through the output of the vxprint command or by using the graphical user interface to look at the status of the disks. You can also see driver error messages on the console or in the system messages file.

If a volume has a disk I/O failure (for example, because the disk has an uncorrectable error), the Volume Manager can 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 can 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 also disabled.

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 identifies the failed plexes. For example, if a disk containing mirrored volumes fails, you can receive mail information as shown in this example:

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

Failures have been detected by the VERITAS Volume Manager:

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

See “Modifying vxrelocd” on page 101 for information on how to send the mail to users other than `root`.

You can determine which disk is causing the failures in the above example message by using this command:

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

The following is a typical output display:

```
                            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 automatically relocates the affected subdisks and initiates any necessary recovery procedures. However, if relocation is not possible or the hot-relocation feature is disabled, you have to investigate the problem and attempt to recover the plexes. These errors can be caused by cabling failures, so check the cables connecting your disks to your system. If there are obvious problems, correct them and recover the plexes with this command:

```
# vxrecover -b home src
```

This command starts recovery of the failed plexes in the background (the command returns before the operation is done). If an error message appears later, or if the plexes become detached again and there are no obvious cabling failures, replace the disk (see “Replacing Disks”).

Complete Disk Failure

If a disk fails completely and hot-relocation is enabled, the mail message lists the disk that failed and all plexes that use the disk. For example, you can receive mail as shown in this example display:

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

Failures have been detected by the VERITAS Volume Manager:

failed disks:
  disk02

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

failing disks:
  disk02
```

This message shows 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 can be a cabling error. If the problem is not a cabling error, replace the disk (see "Replacing Disks" on page 109).

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 can select one of those disks as a replacement.

Note Do not choose the old disk drive as a replacement even though it appears 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, use the following command:

```
# vxinfo
```

Any volumes that are listed as `Unstartable` must be restored from backup. Here is an example `vxinfo` display:

```
home         fsgen     Started
mkting       fsgen     Unstartable
src          fsgen     Started
standvol     gen       Started  (used on some systems)
rootvol      root      Started
swapvol      swap      Started
```

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

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

The `-o bg` option combination resynchronizes plexes as a background task.

If failures are starting to occur on a disk, but the disk has not yet failed completely, replace the disk. This involves two steps:

1. Detach the disk from its disk group.

2. Replace 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 can be affected by the operation are displayed. Here is an example display:

```
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, 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, use this command:

```
# vxassist move mkting !disk02
```

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

After the disk has been removed for replacement, a replacement disk can be specified by selecting item 5 (Replace a failed or removed disk) from the vxdiskadm main menu.

Creating a Disk Group

Disk groups are typically created for a particular set of users or applications. Disks need to be in disk groups before the Volume Manager can use the disks for volumes. Volume Manager always has the default disk group rootdg, but you can add more disk groups if needed.

Note All volumes are created in rootdg if no further specification is given. All commands default to rootdg as well.

To create the disk group newdg, follow these steps:

1. Enter the following command to start vxdiskadd:

   ```
   # vxdiskadd c1t1d0
   ```

2. At the following prompt, press Return to continue:

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

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

   c1t1d0

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

3. At the following prompt, specify the disk group to which the disk should be added (newdg, in this case):

   ```
   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) newdg
   ```

4. When vxdiskadd confirms that no active disk group currently exists with the same name and prompts for confirmation that you want to create this new disk group. Enter **y** to continue:

   ```
   There is no active disk group named newdg.

   Create a new group named newdg? [y,n,q,?] (default: y) y
   ```

5. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

6. When vxdiskadd asks whether this disk should become a hot-relocation spare, enter **n**(or press Return):

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

At the following prompt, enter **n**to indicate that this disk should not be excluded from hot-relocation:

```
Exclude another disk from hot-relocation use? [y,n,q,?]
[default:n] n
```

7. To continue with the task, enter **y** (or press Return) at the following prompt:

```
A new disk group will be created named newdg and the selected
disks will be added to the disk group with default disk names.
c1t1d0

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

   Messages similar to the following should now confirm that this disk is being initialized for Volume Manager use:

```
Initializing device c1t1d0.

Creating a new disk group named newdg containing the disk device
c1t1d0 with the name newdg01.
```

8. To verify that the disk group was created, use the following command:

```
# vxdisk list
```

The Volume Manager returns the following display:

```
DEVICE    TYPE    DISK     GROUP    STATUS

c0t0d0s2  sliced  disk04   rootdg   online

c1t0d0s2  sliced  disk03   rootdg   online

c1t1d0s2  sliced  newdg01  newdg    online
```

Disk groups can also be created by using the operation `vxdg init`. To create a disk group with the `vxdg` utility, use te following command:

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

For example, to create a disk group named `mktdg` on device `c1t0d0s2`, use the following command:

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

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

Upgrading a Disk Group

Prior to the release of Volume Manager 3.0, the disk group version was automatically upgraded (if needed) when the disk group was imported.

The Volume Manager disk group upgrade feature separates the two operations of importing a disk group and upgrading its version. You can import a disk group of down-level version and use it without upgrading it.

When you want to use new features, the disk group can be upgraded. The upgrade is an explicit operation. Once the upgrade occurs, the disk group becomes incompatible with earlier releases of VxVM that do not support the new version.

Before the imported disk group is upgraded, no changes are made to the disk group to prevent its use on the release that it was imported from until the administrator explicitly upgrades it to the current release.

Until completion of the upgrade, the disk group can be used “as is” provided there is no attempt to use the features of the current version. Attempts to use a feature of the current version that is not a feature of the version the disk group was imported from results in an error message similar to this:

```
    vxvm:vxedit: ERROR:  Disk group version doesn’t support feature
```

To use any of the new features, the administrator needs to execute a command to explicitly upgrade the disk group to a version that supports those features.

All disk groups have a version number associated with them. Volume Manager releases support a specific set of disk group versions. Volume Manager can import and perform operations on a disk group of that version. The operations are limited by what features and operations the disk group version supports.

Table 2 summarizes the VxVM releases that introduce and support specific disk group versions, as follows:

Table 2. Disk Group Version Assignments

VxVM Release	Introduces Version	Supports Versions
1.2	10	10
1.3	15	15
2.0	20	20
2.2	30	30
2.3	40	40
3.0	60	20-60
3.1	70	20-70

Note With the exception of VxVM 3.0, all VxVM releases perform the upgrade of supported disk group versions *when* they are imported.

Importing the disk group of a previous version on a VxVM 3.0 system prevents the use of features introduced since that version was released. Table 3 summarizes features not supported by specific disk group versions, as follows:

Table 3. Disk Group Features Not Supported

Disk Group Version	Features Not Supported
40	
30	Hot Relocation
20	VxSmartSync Recovery Accelerator
10, 15	RAID-5 Volumes, New-style Stripes, Recovery Checkpointing, Diskgroup Configuration Copy Limiting, Dirty Region Logging, Mirrored Volumes Logging

You can get a disk group version listing by specifying a disk group name with this command:

```
# vxdg list dgname
```

You can determine the disk group version by using the `vxprint`(1M) command with the `-l` format option.

VxVM upgrades the disk group to the highest version supported by the release of VxVM that is currently running. To upgrade a disk group, use the command:

```
# vxdg upgrade dgname
```

By default, VxVM creates a disk group of the highest version supported by the release of VxVM. For example, VxVM 3.0 creates disk groups of Version 60.

It may be necessary to create a disk group of a back-level version. A disk group created in default fashion on a system running VxVM Release 3.0 would be of disk group version 60. It would not be importable on a system running VxVM Release 2.5, which only supports up to version 50. Therefore, to create a disk group on a system running VxVM Release 3.0 that can be imported by a system running VxVM Release 2.5, the disk group must be created with a version of 40 or less.

To create such a disk group, the `-T` *version* option can be specified to the `vxdg init` command. The *version* should be the disk group version desired for the disk group. For example, to create a disk group that can be imported by a system running VxVM 2.5, use the following command:

```
# vxdg -T 40 init newdg newdg01=c0t3d0s2
```

creates the disk group newdg with a disk group version of 40. This disk group can be imported by VxVM Release 2.5. Note that while the disk group can be imported on the VxVM 2.5 system, attempts to use VxVM 3.0 features fail.

Removing a Disk Group

To remove a disk group, unmount and stop any volumes in the disk group and then use this 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.

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. You do not have to specify the configuration again.

Use the following steps 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 with this command:

   ```
   # vxdg deport diskgroup
   ```

2. Move all the disks to the second system and perform the steps necessary (system-dependent) to make the second system and Volume Manager recognize the new disks.

   This may require a reboot, in which case the vxconfigd daemon is restarted and recognizes the new disks. If you do not reboot, use the command vxdctl enable to restart vxconfigd so Volume Manager also recognizes the disks.

3. Import (enable local access to) the disk group on the second system with this command:

   ```
   # vxdg import diskgroup
   ```

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

   ```
   # vxrecover -g diskgroup -sb
   ```

You can move disks from a system that has crashed. In this case, you cannot 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) are not 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 is corrupted. The disk and its data 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 remain and must be cleared. The system returns the this error message:

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

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

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

It is possible to clear the locks during import by using this command:

```
# vxdg -C import diskgroup
```

Note You must be careful when using the `vxdisk clearimport` or `vxdg -C import` command on systems that do have dual-ported disks. Clearing the locks allows those disks to be accessed at the same time from multiple hosts 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 these error messages is displayed:

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

This message indicates a fatal error that 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 error.

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

```
# vxdg -f import diskgroup
```

Note You must be careful when using the `-f` option. It can cause the same disk group to be imported twice from different sets of disks, causing the disk group to become inconsistent.

These operations can be performed using `vxdiskadm`. To deport a disk group by using `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 prompts to see if you want to clear any that are found. It also starts volumes in the disk group.

Renaming Disk Groups

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

For example, because every system running the Volume Manager must have a single `rootdg` default disk group, importing or deporting `rootdg` across systems is a problem. There cannot be two `rootdg` disk groups on the same system. This problem can be avoided by renaming the `rootdg` disk group during the import or deport.

To give a new name to the disk group during import, use this command:

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

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

A disk group can also be renamed on deport with this command:

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

When renaming on deport, you can specify 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.

To temporarily move the `rootdg` disk group from one host to another (for example, for repair work on the root volume) and then move the disk group back, use the following procedure:

1. On the original host, identify the disk group ID of the `rootdg` disk group to be imported to the other host by using this command:

   ```
   # vxdisk -s list
   ```

   The output display includes disk group information similar to this example:

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

2. On the importing host, import and rename the `rootdg` disk group with this command:

   ```
   # 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 it does not conflict with the existing `rootdg`); and *diskgroup* is the disk group ID of the disk group being imported (for example, `774226267.1025.tweety`).

   If a reboot or crash occurs at this point, the temporarily-imported disk group becomes unimported and requires a reimport.

3. After the necessary work has been done on the imported `rootdg`, deport it back to its original host with this command:

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

   where *hostname* is the name of the system whose `rootdg` is being returned (the system 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 then autoimports its `rootdg` on the next reboot.

Reserving Minor Numbers for Disk Groups

Volume Manager allows you to select a range of minor numbers for a specified disk group. Use this range of numbers during the creation of a volume. This guarantees that each volume has the same minor number across reboots or reconfigurations. If two disk groups have overlapping ranges, an import collision is detected and an avoidance or renumbering mechanism is then needed.

If you allocate volume device numbers in separate ranges for each disk group, all disk groups in a group of machines can be moved without causing device number collisions.

To set a base volume device minor number for a disk group, use the followingcommand:

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

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

If you do not specify a `minor` operand on the `vxdg init` command line, the Volume Manager chooses a random number. The number chosen is at least 1000 or is a multiple of 1000, and yields a usable range of 1000 device numbers. The chosen default number does not overlap within a range of 1000 of any currently imported disk groups. It also does not overlap any currently allocated volume device numbers.

Note The default policy ensures that a small number of disk groups can be merged successfully between a set of machines. However, where disk groups are merged automatically using fail-over mechanisms, you should select ranges that avoid overlap.

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

Destroying a Disk Group

The `vxdg` command provides a destroy option that removes a disk group from the system and frees the disks in that disk group for use in other disk groups. Removeunnecessary disk groups so that the disks can be used by other disk groups using the following command:

```
# vxdg destroy diskgroup
```

The `vxdg deport` command can still be used to make disks inaccessible. The Volume Manager prevents disks in a deported disk group from being used in other disk groups.

Using Special Devices

This section describes special devices the Volume Manager uses to do administrative tasks.

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 the file systems are mounted on volumes instead.

Disk encapsulation requires that 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. (See the *VERITAS Volume Manager Installation Guide* and the vxencap(1M) manual page for more information.)

You can encapsulate a disk that does not have space available for the Volume Manager private region partition. The vxdisk utility encapsulates disks that do not have available space. This is done by using special types of disk devices, called nopriv devices, that do not have private regions.

To use vxdisk, 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 this command:

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

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

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

You create volumes for other partitions on the disk drive by:

- adding the device to a disk group
- determine where those partitions reside within the encapsulation partition
- use vxassist to create a volume with that offset and length

vxassist, by default, reinitializes the data area of a volume that it creates. If there is data to be preserved on the partition, do not use vxassist. Create the volume with vxmake and start the volume with vxvol init active.

A drawback with using the nopriv 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. Because nopriv devices do not have private regions and have no identifying information stored on the physical disk, tracking 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.

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 disk device. Often, the contents of a RAM disk are erased when the system is rebooted. RAM disks that are erased on reboot prevent the Volume Manager from identifying physical disks. This is because information stored on the physical disks (now erased on reboot) is used to identify the disk.

nopriv devices have a special feature to support RAM disks: a *volatile* option which indicates to the Volume Manager that the device contents do not survive reboots. Volatile devices receive special treatment on system startup. If a volume is mirrored, plexes made from volatile devices are always recovered by copying data from nonvolatile 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, use the following command:

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

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

Some 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 using the following command:

```
# 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.

Using vxdisk To Display Multipathing Information

Note This section applies only to systems with the Dynamic Multipathing (DMP) feature.

In Volume Manager, physical disks connected to the system are represented as metadevices with one or more physical access paths. The access paths depend on whether the disk is a single disk or part of a multiported disk array connected to the system. You use the vxdisk utility to display the paths of a metadevice, and to display the status of each path (for example, enabled or disabled). For example, to display details on disk named disk01, use the following command:

```
# vxdisk list disk01
```

The Volume Manager returns this 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
```

The display shows two paths to a physical device represented by the metadevice c2t0d0s2. The path c2t0d0s2 is active and the other path c1t0d0s2 is in a failed state.

vxdiskadm Menu Interface Tasks

This section describes the menu-driven interface known as Volume Manager Support Operations (or vxdiskadm). vxdiskadm is used to perform physical and logical device administration.

This section provides the following information on performing disk and volume tasks using vxdiskadm:

- Bringing Physical Disks Under Volume Manager Control
- Adding a Disk for Future Use
- Reinitializing a Disk for Volume Manager Use
- Adding a VM Disk to the Hot-Relocation Pool
- Removing a VM Disk From the Hot-Relocation Pool
- Displaying Information on Physical Disks
- Removing a Physical Disk
- Disabling a Physical Disk (Taking a Physical Disk Offline)
- Enabling a Physical Disk
- Replacing a Physical Disk
- Replacing a Failed or Removed Disk
- Adding a Disk to a Disk Group
- Creating a Disk Group
- Deporting a Disk Group
- Importing a Disk Group

Starting vxdiskadm

To start vxdiskadm, use the following command:

```
# vxdiskadm
```

The vxdiskadm main menu is displayed.

The vxdiskadm Main Menu

The vxdiskadm main menu is as follows:

```
Volume Manager Support Operations
Menu: VolumeManager/Disk

1    Add or initialize one or more disks
2    Encapsulate one or more disks
3    Remove a disk
4    Remove a disk for replacement
5    Replace a failed or removed disk
```

```
6    Mirror volumes on a disk
7    Move volumes from a disk
8    Enable access to (import) a disk group
9    Remove access to (deport) a disk group
10   Enable (online) a disk device
11   Disable (offline) a disk device
12   Mark a disk as a spare for a disk group
13   Remove (deport) and destroy a disk group
14   Unrelocate subdisks back to a disk
15   Exclude a disk from hot-relocation use
16   Make a disk available for hot-relocation use

list List disk information

?    Display help about menu
??   Display help about the menuing system
q    Exit from menus

Select an operation to perform
```

- **?** can be entered at any time to provide help in using the menu. The output of **?** is a list of operations and a definition of each.
- **??** lists inputs that can be used at any prompt.
- **q** returns you to the main menu if you need to restart a process; however, using **q** at the main menu level exits the Volume Manager Support Operations.

The `vxdiskadm` menu provides access to the following tasks. The numbers correspond to the items listed in the main menu:

1. Add or initialize one or more disks.

   You can add formatted disks to the system. SCSI disks are already formatted. For other disks, see the manufacturer's documentation for formatting instructions. You are prompted for the disk device(s). You can specify the disk group to which the disk(s) should be added; if `none` is selected, the disk is held as a spare to be used for future operations or disk replacements without needing to be initialized at that time. You can also specify that selected disks be marked as hot-relocation spares for a disk group. If the disk has not been initialized already, the disk is partitioned and initialized for use with the Volume Manager.

2. Encapsulate one or more disks.

   You can bring a disk that was added to the system before installing the Volume Manager under Volume Manager control. You are prompted for the disk device(s), disk group, and disk name(s). The disk is added to the specified disk group. The disk is examined to search for partitions that are used for file systems or other purposes. Volumes are created to replace disk partitions as the means of accessing the existing data. If the encapsulation cannot determine the purpose of a partition automatically, you are asked what to do with it. You can choose to replace the partition with a volume, leave the partition alone, or remove the partition.

You must reboot the system if any partitions are being used for mounted file systems or for running applications. You may have to modify application configuration files to use volumes, rather than direct disk devices, to access the disk partitions. File system mount information is adjusted automatically.

3. Remove a disk.

   You can remove a disk from a disk group. You are prompted for the name of a disk to remove. You cannot remove a disk if any volumes use storage on that disk. If any volumes are using storage on the disk, you have the option of asking the Volume Manager to move that storage to other disks in the disk group.

Note You cannot remove the last disk in a disk group using this task. If you wish to use all the remaining disks in a disk group for some purpose, you should disable (deport) the disk group. You will then be able to reuse the disks. `rootdg` cannot, however, be deported.

4. Remove a disk for replacement.

   You can remove a physical disk from a disk group, while retaining the disk name. This changes the state for the named disk to `removed`. If there are any initialized disks that are not part of a disk group, you are given the option of using one of these disks as a replacement.

5. Replace a failed or removed disk.

   You can specify a replacement disk for a disk that you removed with the `Remove a disk for replacement` menu entry, or one that failed during use. You are prompted for a disk name to replace and a disk device to use as a replacement. You can choose an uninitialized disk, in which case the disk will be initialized, or you can choose a disk that you have already initialized using the `Add or initialize a disk` menu operation.

6. Mirror volumes on a disk

   You can mirror volumes on a disk. These volumes can be mirrored to another disk with available space. Creating mirror copies of volumes in this way protects against data loss in case of disk failure. Volumes that are already mirrored or that are comprised of more than one subdisk will not be mirrored with this task. Mirroring volumes from the boot disk will produce a disk that can be used as an alternate boot disk.

7. Move volumes from a disk.

   You can move any volumes (or parts of a volume) that are using a disk onto other disks. Use this menu task immediately prior to removing a disk, either permanently or for replacement.

Note Simply moving volumes off a disk, without also removing the disk, does not prevent other volumes from being moved onto the disk by future operations.

8. Enable access to (import) a disk group.

   You can enable access by this system to a disk group. If you wish to move a disk group from one system to another, you must first disable (deport) it on the original system. Then, move the disks from the deported disk group to the other system and enable (import) the disk group there. You are prompted for the disk group name.

9. Disable access to (deport) a disk group

   You can disable access to a disk group that is currently enabled (imported) by this system. Deport a disk group if you intend to move the disks in a disk group to another system. Also, deport a disk group if you want to use all of the disks remaining in a disk group for some new purpose.

   You are prompted for the name of a disk group. You are asked if the disks should be disabled (offlined). For removable disk devices on some systems, it is important to disable all access to the disk before removing the disk.

10. Enable (online) a disk device.

    If you move a disk from one system to another during normal system operation, the Volume Manager will not recognize the disk automatically. Use this menu task to tell the Volume Manager to scan the disk to identify it, and to determine if this disk is part of a disk group. Also, use this task to re-enable access to a disk that was disabled by either the disk group deport task or the disk device disable (offline) operation.

11. Disable (offline) a disk device.

    You can disable all access to a disk device through the Volume Manager. This task can be applied only to disks that are not currently in a disk group. Use this task if you intend to remove a disk from a system without rebooting.

    Note that some systems do not support disks that can be removed from a system during normal operation. On those systems, the offline operation is seldom useful.

12. Mark a disk as a spare for a disk group.

    You can reserve a disk as an automatic replacement disk (for hot-relocation) in case another disk in the disk group should fail.

13. Remove (deport) and destroy a disk group.

    You can remove and destroy a disk group.

14. Unrelocate subdisks back to disk.

    VxVM Hot-relocation allows the system to automatically react to I/O failures on a redundant VxVM object at the subdisk level and take necessary action to make the object available again. This mechanism detects I/O failures in a subdisk, relocates the subdisk, and recovers the plex associated with the subdisk. After the disk has been replaced, Volume Manager provides the `vxunreloc` utility, which can be used to restore the system to the same configuration that existed before the disk failure. `vxunreloc` allows you to move the hot-relocated subdisks back onto a disk that was replaced due to a disk failure.

15. Exclude a disk from hot-relocation use.

    Exclude disks in the free pool (non-spares) to be used by hot-relocation.

16. Make a disk available for hot-relocation use.

    Undo Step 15, and make disks in the free pool (non-spares) available for hot-relocation.

When performing disk administration, it is important that you recognize the difference between a *device name* and a *disk name.*

Note Your system may use a device name that differs from the examples. See "Understanding Volume Manager" on page 17 for more information on device names.

The *device name* (sometimes referred to as *devname* or *disk access name*) is the location of the disk. The syntax of a device name is `c#b#t#d#s#`, where:

- `c#` is the number of the controller to which the disk drive is attached.
- `b#` is the corresponding bus number (if used on your system)
- `t#` is the number of the target disk on that controller.
- `d#` is the number of the disk.
- `s#` is the number of the disk slice.

The full pathname of a device is `/dev/vx/dmp/`*`devicename`*. In this document, only the device name is listed and `/dev/vx/dmp` is assumed. An example of a device name is `c0t0d0s2`.

The *disk name* (sometimes referred to as *disk media name*) is an administrative name for the disk, such as `disk01`. If you do not assign a disk name, the disk name defaults to `disk`*##* if the disk is being added to `rootdg` (where *##* is a sequence number). Otherwise, the default disk name is `groupname`*##*, where *groupname* is the name of the disk group to which the disk is added.

Bringing Physical Disks Under Volume Manager Control

When you add a disk to a system that is running Volume Manager, you need to put the disk under Volume Manager control so that the Volume Manager can control the space allocation on the disk.

Unless another disk group is specified, Volume Manager places new disks in the default disk group, `rootdg`. Instructions on creating additional disk groups are provided later in this chapter.

The method by which you place a disk under Volume Manager control depends on the circumstances:

- If the disk is new, it needs to be initialized and placed under Volume Manager control (see “Placing a Disk Under Volume Manager Control” on page 129).
- If the disk is not needed immediately, it can be initialized (but not added to a disk group) and reserved for future use (see “Adding a Disk for Future Use” on page 139).
- If the disk was previously initialized for future Volume Manager use, it can be reinitialized and placed under Volume Manager control (see “Reinitializing a Disk for Volume Manager Use” on page 139).
- If the disk was previously in use, but not under Volume Manager control, you may wish to preserve existing data on the disk while still letting Volume Manager take control of the disk. This can be accomplished using encapsulation (see “Encapsulating a Disk for Volume Manager Use” on page 136).
- Multiple disks on one or more controllers can be placed under Volume Manager control simultaneously. Depending on the circumstances, all of the disks may not be processed the same way (see “Placing Multiple Disks Under Volume Manager Control” on page 132).

  When initializing or encapsulating multiple disks at once, it is possible to exclude certain disks or certain controllers. To exclude disks, list the names of the disks to be excluded in the file `/etc/vx/disks.exclude` before the initialization or encapsulation. Similarly, you can exclude all disks on specific controllers from initialization or encapsulation by listing those controllers in the file `/etc/vx/cntrls.exclude`.

The sections that follow provide detailed examples of how to use `vxdiskadm` to place disks under Volume Manager control in various ways and circumstances.

Note A disk must be formatted (using the `format` command, for example) or added to the system (using `diskadd`) before it can be placed under Volume Manager control.

If you attempt to place an unformatted disk under Volume Manager control through `vxdiskadm`, the initialization begins as normal, but quits with a message that informs you that the disk does not appear to be valid and may not be formatted. If this happens, you need to format the disk properly and then attempt to place the disk under Volume Manager control again.

Placing a Disk Under Volume Manager Control

This section describes how to place a formatted disk under Volume Manager control. The disk can be new or previously used outside Volume Manager.

Note Initialization does not preserve data on disks.

Initialize a single disk for Volume Manager use as follows:

1. Select menu item `1 (Add or initialize one or more disks)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the disk device name of the disk to be added to Volume Manager control (or enter **`list`** for a list of disks):

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

Use this operation to add one or more disks to a disk group. You
can add the selected disks to an existing disk group or to a new
disk group that will be created as a part of the operation. The
selected disks may also be added to a disk group as spares. The
selected disks may also be initialized without adding them to a
disk group leaving the disks available for use as replacement
disks.

More than one disk or pattern may be entered at the prompt. Here
are some disk selection examples:

all:            all disks
c3 c4t2:        all disks on both controller 3 and controller
                4,target 2
c3t4d0:         a single disk

Select disk devices to add:
[<pattern-list>,all,list,q,?] list
```

<*pattern-list*> can be a single disk, or a series of disks and / or controllers (with optional targets). If <*pattern-list*> consists of multiple items, those items must be separated by white space.

If you enter **`list`** at the prompt, `vxdiskadm` displays a list of the disks available to the system, followed by a prompt at which you should enter the device name of the disk to be added:

```
DEVICE       DISK         GROUP        STATUS
c0t0d0       disk01       rootdg       online
c0t1d0       disk02       rootdg       online
c0t2d0       disk03       rootdg       online
c0t3d0       -            -            online
c1t0d0       disk10       rootdg       online
c1t0d1       -            -            error
.
.
.
c3t0d0       -            -            error
c3t1d0       disk33       rootdg       online
c3t2d0       disk34       rootdg       online
c3t3d0       disk35       rootdg       online

Select disk devices to add:
[<pattern-list>,all,list,q,?] c1t0d1
```

All disks attached to the system are recognized by the Volume Manager and displayed here.

The word `error` in the `STATUS` line tells you that a disk has not yet been added to Volume Manager control. These disks may or may not have been initialized before. The disks that are listed with a disk name and disk group cannot be used for this task, as they are already under Volume Manager control.

3. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

   c1t2d0

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

4. At the following prompt, specify the disk group to which the disk should be added or press Return to accept `rootdg`:

```
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)
```

5. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

6. When `vxdiskadm` asks whether this disk should become a hot-relocation spare, enter **n**(or press Return):

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

7. When `vxdiskadm` asks whether to exclude this disk from hot-relocation use, enter **n** (or press Return):

   ```
   Exclude disk from hot-relocation use? [y,n,q,?] (default: n)
   ```

8. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

   ```
   c1t2d0
   ```

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

9. If you are sure that there is no data on this disk, enter **n** to avoid encapsulation. When `vxdiskadm` asks if you want to initialize the disk instead, enter **y**:

   ```
   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]
   ```

   ```
   c1t2d0
   ```

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

   ```
   c1t2d0
   ```

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

Messages similar to the following should now confirm that disk `c1t2d0` is being placed under Volume Manager control. On some systems, you are also given the option of performing surface analysis.

```
Initializing device c1t2d0.

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

 Adding disk device c1t2d0 to disk group rootdg with disk
 name disk39.
```

10. At the following prompt, indicate whether you want to continue to initialize more disks (**y**) or return to the `vxdiskadm` main menu (**n**):

```
Add or initialize other disks? [y,n,q,?] (default: n)
```

Placing Multiple Disks Under Volume Manager Control

This section describes how to place multiple disks under Volume Manager control simultaneously. The set of disks can consist of all disks on the system, all disks on a controller, selected disks, or a combination thereof.

Depending on the circumstances, all of the disks may not be processed the same way. For example, some may be initialized, while others may be encapsulated.

Note Initialization does not preserve data on disks.

When initializing or encapsulating multiple disks at one time, it is possible to exclude certain disks or certain controllers. To exclude disks, list the names of the disks to be excluded in the file `/etc/vx/disks.exclude` before the initialization or encapsulation. You can exclude all disks on specific controllers from initialization or encapsulation by listing those controllers in the file `/etc/vx/cntrls.exclude`.

Place multiple disks under Volume Manager control at one time as follows:

1. Select menu item `1 (Add or initialize one or more disks)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the *pattern-list* for the disks to be added to Volume Manager control. In this case, enter **`c3`** to indicate all disks on controller 3:

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

Use this operation to add one or more disks to a disk group.
You can add the selected disks to an existing disk group or to
a new disk group that will be created as a part of the
operation. The selected disks may also be added to a disk
```

```
group as spares. The selected disks may also be initialized
without adding them to a disk group leaving the disks
available for use as replacement disks.

More than one disk or pattern may be entered at the prompt.
Here are some disk selection examples:

all:            all disks
c3 c4t2:        all disks on both controller 3 and controller
                4,target 2
c3t4d0:         a single disk

Select disk devices to add:
[<pattern-list>,all,list,q,?] c3
```

where *<pattern-list>* can be a single disk, or a series of disks and / or controllers (with optional targets). If *<pattern-list>* consists of multiple items, those items must be separated by white space.

If you do not know the address (device name) of the disk to be added, enter **`list`** at the prompt for a complete listing of available disks.

3. To continue the operation, enter **y** (or press Return) at the following prompt:

```
Here are the disks selected.  Output format: [Device_Name]

c3t0d0 c3t1d0 c3t2d0 c3t3d0

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

4. To add these disks to the default disk group, `rootdg`, enter **y** (or press Return) at the following prompt:

```
You can choose to add these disks to an existing disk group, a new
disk group, or you can leave these disks 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
disks available for future use, specify a disk group name of
“none”.

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

5. To allow `vxdiskadm` to use default disk names for each of the disks, enter **y** (or Press Return) at the following prompt:

```
Use default disk names for these disks? [y,n,q,?] (default: y) y
```

6. At the following prompt, enter **n**to indicate that these disks should not be used as hot-relocation spares:

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

7. When vxdiskadm asks whether to exclude this disk from hot-relocation use, enter **n** (or press Return):

   ```
   Exclude disk from hot-relocation use? [y,n,q,?] (default: n)
   ```

8. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

   c3t0d0 c3t1d0 c3t2d0 c3t3d0

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

9. The following prompt lists any disks that have already been initialized for Volume Manager use; enter **Y** to indicate that all of these disks should be used now:

   ```
   The following disk devices appear to have been initialized already.
   The disks are currently available as replacement disks.
   Output format: [Device_Name]

   c3t1d0 c3t2d0 c3t3d0

   Use these devices? [Y,N,S(elect),q,?] (default: Y) Y
   ```

   Note that this prompt allows you to indicate "yes" or "no" for *all* of these disks (**Y** or **N**) or to select how to process each of these disks on an individual basis (**S**).

   If you are sure that you want to reinitialize all of these disks, enter **Y** at the following prompt:

   ```
   The following disks you selected for use appear to already have
   been initialized for the Volume Manager.  If you are certain the
   disks already have been initialized for the Volume Manager, then
   you do not need to reinitialize these disk devices.
   Output format: [Device_Name]

   c3t1d0 c3t2d0 c3t3d0

   Reinitialize these devices? [Y,N,S(elect),q,?] (default: Y) Y
   ```

10. vxdiskadm now indicates that one of the disks on controller 3 is a candidate for encapsulation; enter **y** (or press Return) to encapsulate this disk:

```
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]

c3t0d0

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

Encapsulation allows you to add an active disk to Volume Manager control and preserve the data on that disk.

Note Disk encapsulation requires that you reboot the system and may require a few subsequent reboots. You will be prompted for these operations, as necessary.

`vxdiskadm` now confirms those disks that are being initialized and added to Volume Manager control with messages similar to the following:

```
Initializing device c3t1d0.

Initializing device c3t2d0.

Initializing device c3t3d0.

Adding disk device c3t1d0 to disk group rootdg with disk
name disk33.

Adding disk device c3t2d0 to disk group rootdg with disk
name disk34.

Adding disk device c3t3d0 to disk group rootdg with disk
name disk35.
```

In addition to the output displayed above, you may see prompts that give you the option of performing surface analysis.

11. `vxdiskadm` then confirms any disks that have been selected for encapsulation and prompts you for permission to proceed with the encapsulation; enter **y** (or press Return) to continue encapsulation:

```
The following disk has been selected for encapsulation.
Output format: [Device_Name]

c3t0d0

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

`vxdiskadm` now displays an encapsulation status and informs you that you must perform a shutdown and reboot as soon as possible:

```
The disk device c3t0d0 will be encapsulated and added to the
disk group rootdg with the disk name disk38.

The first stage of encapsulation has completed successfully.
You should now reboot your system at the earliest possible
opportunity.

The encapsulation will require two or three reboots which will
happen automatically after the next reboot.  To reboot execute
the command:

shutdown -g0 -y -i6

This will update the /etc/vfstab file so that volume devices are
used to mount the file systems on this disk device.  You will
need to update any other references such as backup scripts,
databases, or manually created swap devices.
```

12. At the following prompt, indicate whether you want to continue to initialize more disks (**y**) or return to the `vxdiskadm` main menu (**n**):

```
Add or initialize other disks? [y,n,q,?] (default: n)
```

Encapsulating a Disk for Volume Manager Use

This section describes how to encapsulate a disk for Volume Manager use. Encapsulation preserves any existing data on the disk when the disk is placed under Volume Manager control.

To reduce the chance of encapsulation failure, make sure that the disk:

- has a small amount of free space (at the beginning or end of the disk) that does not belong to any partition
- has two free partitions
- has an `s2` (or `s0` on some systems) slice that represents the whole disk

On some systems, when encapsulating the boot (root) disk, the swap partition should be tagged as `swap` so that it's possible to dump to that partition later.

Before encapsulating your boot disk, the EEPROM variable `use-nvramrc?` must be set to `true` if you want to take advantage of the Volume Manager boot disk aliases to identify the mirror of the boot disk if a replacement is needed. If this variable is set to `false`, you need to determine which disks are bootable yourself. You can set this variable to `true` as follows:

```
eeprom use-nvramrc?=true
```

To encapsulate a disk for Volume Manager use, use the following procedure:

1. Select menu item `2 (Encapsulate one or more disks)` from the `vxdiskadm` main menu.

Note Your system may use a device name that differs from the examples. See the *VERITAS Volume Manager Getting Started Guide* for more information on device names.

2. At the following prompt, enter the disk device name for the disks to be encapsulated:

```
Encapsulate one or more disks
Menu: VolumeManager/Disk/Encapsulate

Use this operation to convert one or more disks to use the
Volume Manager.This adds the disks to a disk group and
replaces existing partitions with volumes.  Disk encapsulation
requires a reboot for the changes to take effect.

More than one disk or pattern may be entered at the prompt.
Here are some disk selection examples:

all:            all disks
c3 c4t2:        all disks on both controller 3 and controller
                4,target 2
c3t4d0:         a single disk

Select disk devices to encapsulate:
[<pattern-list>,all,list,q,?] c2t5d0
```

Where <*pattern-list*> can be a single disk, or a series of disks and / or controllers (with optional targets). If <*pattern-list*> consists of multiple items, those items must be separated by white space.

If you do not know the address (device name) of the disk to be encapsulated, enter **`l`** or **`list`** at the prompt for a complete listing of available disks.

3. To continue the operation, enter **y** (or press Return) at the following prompt:

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

c2t5d0

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

4. To add the disk to the default disk group, `rootdg`, press Return at the following prompt:

```
You can choose to add this disk to an existing disk group or to
a new disk group.  To create a new disk group, select a disk
group name that does not yet exist.

Which disk group [<group>,list,q,?] (default: rootdg)
```

5. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

6. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

   c2t5d0

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

7. To confirm that encapsulation should proceed, enter **y** (or press Return) at the following prompt:

   ```
   The following disk has been selected for encapsulation.  Output
   format: [Device_Name]

   c2t5d0

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

   A message similar to the following confirms that the disk is being encapsulated for Volume Manager use and tells you that a reboot is needed:

   ```
   The disk device c2t5d0 will be encapsulated and added to the
   disk group rootdg with the disk name disk01.

   The c2t5d0 disk has been configured for encapsulation.

   The first stage of encapsulation has completed successfully.
   You should now reboot your system at the earliest possible
   opportunity.

   The encapsulation will require two or three reboots which
   will happen automatically after the next reboot.  To reboot
   execute the command:

   shutdown -g0 -y -i6

   This will update the /etc/vfstab file so that volume devices
   are used to mount the file systems on this disk device.  You
   will need to update any other references such as backup
   scripts, databases, or manually created swap devices.
   ```

8. At the following prompt, indicate whether you want to encapsulate more disks (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Encapsulate other disks? [y,n,q,?] (default: n) n
   ```

Under some circumstances, encapsulation of a disk can fail. Encapsulation often fails because there is not enough free space available on the disk to accommodate Volume Manager. If this happens, the procedure outlined above ends abruptly with an error message similar to the following:

```
The disk device c2t5d0 will be encapsulated and added to the
disk group rootdg with the disk name disk01.

The encapsulation operation failed with the following error:

It is not possible to encapsulate c2t5d0, for the following
reason:
<vxvm:vxslicer: ERROR: Unsupported disk layout.>

Hit RETURN to continue.
```

Adding a Disk for Future Use

If you want to add a disk to Volume Manager control for future use, follow the steps outlined in “Bringing Physical Disks Under Volume Manager Control” on page 128. However, when you are asked to name a disk group, enter **none** instead of selecting `rootdg` or typing in a disk group name. The disk is then initialized as before, but is reserved for use at a later time. It cannot be used until it is added to a disk group.

Note Do not confuse this type of “spare disk” with a hot-relocation spare disk.

Reinitializing a Disk for Volume Manager Use

This section describes how to reinitialize a disk that has previously been initialized for Volume Manager use.

If the disk you want to add has been used before, but not with Volume Manager, use *either* of the following procedures:

- Encapsulate the disk and preserve its information (see “Encapsulating a Disk for Volume Manager Use” on page 136).
- Reinitialize the disk, allowing the Volume Manager to configure the disk for Volume Manager. Note that reinitialization does not preserve data on the disk. If you wish to have the disk reinitialized, make sure that the disk does not contain data that should be preserved.

To reinitialize a disk for Volume Manager use, use the following procedure:

1. Select menu item `1 (Add or initialize one or more disks)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the disk device name of the disk to be added to Volume Manager control:

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

Use this operation to add one or more disks to a disk group.
You can add the selected disks to an existing disk group or to
a new disk group that will be created as a part of the
operation. The selected disks may also be added to a disk group
as spares. The selected disks may also be initialized without
adding them to a disk group leaving the disks available for use
as replacement disks.

More than one disk or pattern may be entered at the prompt. Here
are some disk selection examples:

all:            all disks
c3 c4t2:        all disks on both controller 3 and controller
                4,target 2
c3t4d0:         a single disk

Select disk devices to add:
[<pattern-list>,all,list,q,?] c1t3d0
```

   Where <*pattern-list*> can be a single disk, or a series of disks and/or controllers (with optional targets). If <*pattern-list*> consists of multiple items, those items must be separated by white space.

   If you do not know the address (device name) of the disk to be added, enter **`l`** or **`list`** at the prompt for a complete listing of available disks.

3. To continue with the operation, enter **`y`** (or press Return) at the following prompt:

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

c1t3d0

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

4. At the following prompt, specify the disk group to which the disk should be added or press Return to accept `rootdg`:

```
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)
```

5. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

6. When `vxdiskadm` asks whether this disk should become a hot-relocation spare, enter **n**(or press Return):

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

7. When `vxdiskadm` asks whether to exclude this disk from hot-relocation use, enter **n**(or press Retrun):

   ```
   Exclude disk from hot-relocation use? [y,n,q,?] (default: n)
   ```

8. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

   c1t3d0

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

9. The following prompt indicates that this disk has been previously initialized for future Volume Manager use; enter **y** to confirm that you now want to use this disk:

   ```
   The following disk device appears to have been initialized
   already. The disk is currently available as a replacement disk.
   Output format: [Device_Name]

   c1t3d0

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

10. To reinitialize the disk, enter **y** (or press Return) at the following prompt:

    ```
    The following disk you selected for use appears to already have
    been initialized for the Volume Manager.  If you are certain the
    disk has already been initialized for the Volume Manager, then
    you do not need to reinitialize the disk device.
    ```

```
Output format: [Device_Name]
```

```
c1t3d0
```

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

Messages similar to the following now confirm that this disk is being reinitialized for Volume Manager use:

```
Initializing device c1t3d0.
```

```
Adding disk device c1t3d0 to disk group rootdg with disk
name disk40.
```

11. At the following prompt, indicate whether you want to continue to initialize more disks (**y**) or return to the `vxdiskadm` main menu (**n**):

    ```
    Add or initialize other disks? [y,n,q,?] (default: n)
    ```

Adding a VM Disk to the Hot-Relocation Pool

Hot-relocation allows the system to automatically react to I/O failure by relocating redundant subdisks to other disks. Hot relocation then restores the affected Volume Manager objects and data. If a disk has already been designated as a spare in the disk group, the subdisks from the failed disk are relocated to the spare disk. Otherwise, any suitable free space in the disk group is used except for the free space on the disks that were previously excluded from hot-relocation use.

To designate a disk as a spare, do the following:

1. Select menu item `12 (Mark a disk as a spare for a disk group)` from the `vxdiskadm` main menu.

2. At the following prompt, enter a disk name (such as `disk01`):

    ```
    Mark a disk as a spare for a disk group
    Menu: VolumeManager/Disk/MarkSpareDisk
    ```

    ```
    Use this operation to mark a disk as a spare for a disk group.
    This operation takes, as input, a disk name. This is the same
    name that you gave to the disk when you added the disk to the
    disk group.
    ```

    ```
    Enter disk name [<disk>,list,q,?] disk01
    ```

    `vxdiskadm` displays the following confirmation:

    ```
    Marking of disk01 in rootdg as a spare disk is complete.
    ```

3. At the following prompt, indicate whether you want to add more disks as spares (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Mark another disk as a spare? [y,n,q,?] (default: n)
   ```

   Any VM disk in this disk group can now use this disk as a spare in the event of a failure. If a disk fails, hot-relocation should automatically occur (if possible). You should be notified of the failure and relocation through electronic mail. After successful relocation, you may want to replace the failed disk.

   Free space is used automatically by hot-relocation in case spare space is not sufficient to relocate failed subdisks. The user can limit this free space usage by hot-relocation by specifying which free disks should not be touched by hot-relocation.

Removing a VM Disk From the Hot-Relocation Pool

While a disk is designated as a spare, the space on that disk is not used as free space for the creation of Volume Manager objects within its disk group. If necessary, you can free a spare disk for general use by removing it from the pool of hot-relocation disks.

To verify which disks are currently designated as spares, select the `list` menu item from the `vxdiskadm` main menu. Disks that are spares are listed with the `spare` flag.

To remove a disk from the hot-relocation pool, do the following:

1. Select menu item `13 (Turn off the spare flag on a disk)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the name of a spare disk (such as `disk01`):

   ```
   Turn off the spare flag on a disk
   Menu: VolumeManager/Disk/UnmarkSpareDisk

   Use this operation to turn off the spare flag on a disk.
   This operation takes, as input, a disk name. This is the same
   name that you gave to the disk when you added the disk to the
   disk group.

   Enter disk name [<disk>,list,q,?] disk01
   ```

   `vxdiskadm` displays the following confirmation:

   ```
   Disk disk01 in rootdg no longer marked as a spare disk.
   ```

3. At the following prompt, indicate whether you want to disable more spare disks (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Turn-off spare flag on another disk? [y,n,q,?] (default: n)
   ```

Moving Hot-relocate Subdisks Back to a Disk

This option can be used to move the hot-relocated subdisks back to the disk where they originally resided after it has been replaced due to a disk failure.

1. Select menu item 14 (Move hot-relocated subdisks back to a disk) from the vxdiskadm main menu.

2. This option prompts for the original disk media name first.

   Enter the disk media name where the hot-relocated subdisks originally resided at the following prompt:

   ```
   Enter the original disk name [<disk>,list,q,?]
   ```

However, if there are no hot-relocated subdisks in the system, vxdiskadm will print, `Currently there are no hot-relocated disks` and ask you to press RETURN to continue.

3. The option then prompts for a new destination disk.

   ```
   While unrelocating the subdisks, you can choose to move the
   subdisks to a different disk from the original disk.
   Unrelocate to a new disk [y,n,q,?] (default: n)
   ```

4. If moving subdisks to the original offsets is not possible, you can also choose the "force option" to unrelocate the usbdisks to the specifed disk but not necessarily to the exact original offsets.

   ```
   Use -f option to unrelocate the subdisks if moving to the exact
   offset fails? [y,n,q,?] (default: n)
   ```

5. This option then confirms the requested operation and asks if continuing with the operation is desired.

   ```
   Requested operation is to move all the subdisks which were
   hot-relocated from disk10 back to disk10 of disk group rootdg.
   Continue with operation? [y,n,q,?] (default: y)
   ```

6. At the end, this option informs the user of the status of the operation.

   ```
   Unrelocate to disk disk10 is complete.
   ```

Excluding a Disk From Hot-Relocation Use

To exclude a disk from hot-relocation use:

1. Select menu item 15 (Exclude a disk from hot-relocation use) from the vxdiskadm main menu.

2. At the following prompt, enter disk name (such as disk01):

   ```
   Exclude a disk from hot-relocation use
   Menu: VolumeManager/Disk/MarkNoHotUse
   ```

   Use this operation to exclude a disk from hot-relocation use. This operation takes, as input, a disk name. This is the same name that you gave to the disk when you added the disk to the disk group.

   ```
   Enter disk name [<disk>,list,q,?] disk01
   ```

   vxdisdkadm displays the following confirmation:

   ```
   Excluding disk01 in rootdg from hot-relocation use is complete.
   ```

3. At the following prompt, indicate whether you want to add more disks to be excluded from hot-relocation (**y**) or press Return to the vxdiskadm main menu(**n**):

   ```
   Exclude another disk from hot-relocation use? [y,n,q,?]
   (default: n)
   ```

Making a Disk Available for Hot-Relocation Use

If a disk was previously excluded from hot-relocation use, the following will undo and add the disk back to the hot-relocation pool:

To make a disk available for hot-relocation use:

1. Select menu item 16 (Make a disk available for hot-relocation use) from the vxdiskadm main menu.

   ```
   Make a disk available for hot-relocation use
   ```

2. At the following prompt, enter disk name (such as disk01):

   ```
   Make a disk available for hot-relocation use
   Menu: VolumeManager/Disk/UnmarkNoHotUse
   ```

   Use this operation to make a disk available for hot-relocation use. This only applies to disks that were previously excluded from hot-relocation use.

   This operation takes, as input, a disk name. This is the same name that you gave to the disk when you added the disk to the disk group.

   ```
   Enter disk name [<disk>,list,q,?] disk01
   ```

   vxdiskadm displays the following confirmation:

```
Making disk01 in rootdg available for hot-relocation use is
complete.
```

3. At the following prompt, indicate whether you want to make more disks available for hot-relocation (**y**) or return to the `vxdiskadm` main menu(**n**):

   ```
   Make another disk available for hot-relocation use? [y,n,q,?]
   (default: n)
   ```

Displaying Information on Physical Disks

Displaying disk information shows you which disks are initialized, to which disk groups they belong, and the disk status. The `list` command displays device names for all recognized disks, the disk names, the disk group names associated with each disk, and the status of each disk.

To display disk information, do the following:

1. Select `list` from the `vxdiskadm` main menu.

2. At the following display, enter the address of the disk you want to see or enter **`all`** for a list of all disks:

   ```
   List disk information
   Menu: VolumeManager/Disk/ListDisk

   Use this menu operation to display a list of disks. You can
   also choose to list detailed information about the disk at
   a specific disk device address.

   Enter disk device or "all" [<address>,all,q,?] (default: all)

   If you enter all, the Volume Manager displays the following
   information (some systems may display rootdisk in place of
   c0t0d0s2):

   DEVICE    DISK       GROUP     STATUS
   c0t0d0    c0t0d0s2   rootdg    online
   c1t0d0    disk01     rootdg    online
   c1t1d0    -          -         online

   Device to list in detail [<address>,none,q,?] (default: none)
   ```

   - If you enter the address of the device for which you want information, complete disk information (including the device name, the type of disk, and information about the public and private partitions of the disk) is shown.

   Once you have examined this information, press Return to return to the main menu.

Removing a Physical Disk

Before removing a disk, make sure it contains no data, all data no longer needed, or the data can be moved to other disks. Then remove the disk as follows:

1. Select menu item 3 (Remove a disk) from the vxdiskadm main menu.

Note You must disable the disk group before you can remove the last disk in that group.

2. At the following prompt, enter the disk name of the disk to be removed:

   ```
   Remove a disk
   Menu: VolumeManager/Disk/RemoveDisk

   Use this operation to remove a disk from a disk group. This
   operation takes a disk name as input. This is the same name
   that you gave to the disk when you added the disk to the disk
   group.

   Enter disk name [<disk>,list,q,?] disk01
   ```

3. If there are any volumes on the disk, the Volume Manager asks you whether they should be evacuated from the disk. If you wish to keep the volumes, answer **y**. Otherwise, answer **n**.

4. At the following verification prompt, press Return to continue:

   ```
   Requested operation is to remove disk disk01 from group rootdg.

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

   The vxdiskadm utility removes the disk from the disk group and displays the following success message:

   ```
   Removal of disk disk01 is complete.
   ```

   You can now remove the disk or leave it on your system as a replacement.

5. At the following prompt, indicate whether you want to remove other disks (**y**) or return to the vxdiskadm main menu (**n**):

   ```
   Remove another disk? [y,n,q,?] (default: n)
   ```

Disabling a Physical Disk (Taking a Physical Disk Offline)

If a disk is corrupted, you must take it offline and remove it. You may be moving the physical disk device to another location to be connected to another system. To take a disk offline, first remove it from its disk group, then do the following:

1. Select menu item `11 (Disable (offline) a disk device)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the address of the disk you want to disable:

   ```
   Disable (offline) a disk device
   Menu: VolumeManager/Disk/OfflineDisk

   Use this menu operation to disable all access to a disk device
   by the Volume Manager. This operation can be applied only to
   disks that are not currently in a disk group. Use this operation
   if you intend to remove a disk from a system without rebooting.

   NOTE: Many systems do not support disks that can be removed from
         a system during normal operation. On such systems, the
         offline operation is seldom useful.

   Select a disk device to disable [<address>,list,q,?] c1t1d0
   ```

   The `vxdiskadm` program disables the specified disk.

3. At the following prompt, indicate whether you want to disable another device (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Disable another device? [y,n,q,?] (default: n)
   ```

Enabling a Physical Disk

If you move a disk from one system to another during normal system operation, the Volume Manager does not recognize the disk automatically. Use the enable disk task to get Volume Manager to scan the disk to identify it, and to determine if this disk is part of a disk group. Also, use this task to re-enable access to a disk that was disabled by either the disk group deport task or the disk device disable (offline) task. To enable a disk, do the following:

1. Select menu item `10 (Enable (online) a disk device)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the device name of the disk to be enabled (or enter **`list`** for a list of devices):

   ```
   Enable (online) a disk device
   Menu: VolumeManager/Disk/OnlineDisk

   Use this operation to enable access to a disk that was disabled
   with the “Disable (offline) a disk device” operation.

   You can also use this operation to re-scan a disk that may have
   been changed outside of the Volume Manager. For example, if a
   ```

```
disk is shared between two systems, the Volume Manager running
on the other system may have changed the disk. If so, you can
use this operation to re-scan the disk.

NOTE: Many vxdiskadm operations re-scan disks without user
      intervention. This will eliminate the need to online a
      disk directly, except when the disk is directly offlined.

Select a disk device to enable [<address>,list,q,?] c1t1d0
```

`vxdiskadm` enables the specified device.

3. At the following prompt, indicate whether you want to enable another device (**y**) or return to the `vxdiskadm` main menu (**n**):

```
Enable another device? [y,n,q,?] (default: n)
```

Note `vxdctl enable` should be issued after a hot disk swap to enable VxVM to recognize the disk and the paths.

Replacing a Physical Disk

If a disk fails, you need to replace that disk with another. This task requires disabling and removing the failed disk and installing a new disk in its place. To replace a disk, do the following:

1. Select menu item 4 `(Remove a disk for replacement)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the name of the disk to be replaced (or enter **`list`** for a list of disks):

```
Remove a disk for replacement
Menu: VolumeManager/Disk/RemoveForReplace

Use this menu operation to remove a physical disk from a disk
group, while retaining the disk name. This changes the state
for the disk name to a removed disk. If there are any

initialized disks that are not part of a disk group, you will be
given the option of using one of these disks as a replacement.

Enter disk name [<disk>,list,q,?] disk02
```

Additional displays show any volumes associated with the disk you wish to remove. You must decide whether to keep the data associated with the volumes or to allow that data to be lost when the disk is replaced. Answer any prompts accordingly.

3. At the following prompt, either select the device name of the replacement disk (from the list provided) or press Return to choose the default disk:

```
The following devices are available as replacements:
c1t1d0

You can choose one of these disks now, to replace disk02.
Select “none” if you do not wish to select a replacement disk.

Choose a device, or select “none”
[<device>,none,q,?] (default: c1t1d0)
```

4. At the following prompt, press Return to continue:

```
Requested operation is to remove disk disk02 from group rootdg.
The removed disk will be replaced with disk device c1t1d0.

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

vxdiskadm displays the following success messages:
Removal of disk disk02 completed successfully.

Proceeding to replace disk02 with device c1t1d0.


Disk replacement completed successfully.
```

5. At the following prompt, indicate whether you want to remove another disk (**y**) or return to the `vxdiskadm` main menu (**n**):

```
Remove another disk? [y,n,q,?] (default: n)
```

Replacing a Failed or Removed Disk

Disks can be removed and then replaced later. To do this, use menu item `4 (Remove a disk for replacement)` to remove a disk, then do the following:

1. Select menu item `5 (Replace a failed or removed disk)` from the `vxdiskadm` main menu.

2. Select the disk name of the disk to be replaced:

```
Replace a failed or removed disk
Menu: VolumeManager/Disk/ReplaceDisk

Use this menu operation to specify a replacement disk for a disk
that you removed with the “Remove a disk for replacement” menu
operation, or that failed during use. You will be prompted for
a disk name to replace and a disk device to use as a replacement.
You can choose an uninitialized disk, in which case the disk
```

```
will be initialized, or you can choose a disk that you have
already initialized using the Add or initialize a disk menu
operation.
```

```
Select a removed or failed disk [<disk>,list,q,?] disk02
```

3. `vxdiskadm` displays the device names of the disk devices available for use as replacement disks; enter the device name of the device of your choice or press Return to select the default device:

Note Your system may use a device name that differs from the examples. See "Understanding Volume Manager" on page 17 for more information on device names.

```
The following devices are available as replacements:
c1t0d0s2 c1t1d0s2
```

```
You can choose one of these disks to replace disk02.
Choose "none" to initialize another disk to replace disk02.
```

```
Choose a device, or select "none"
[<device>,none,q,?] (default: c1t0d0s2)
```

4. At the following prompt, press Return to replace the disk:

```
The requested operation is to use the initialized device
c1t0d0s2 to replace the removed or failed disk disk02 in disk
group rootdg.
```

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

The `vxdiskadm` program displays the following success message:

```
Replacement of disk disk02 in group rootdg with disk device
c1t0d0s2 completed successfully.
```

5. At the following prompt, indicate whether you want to replace another disk (**y**) or return to the `vxdiskadm` main menu (**n**):

```
Replace another disk? [y,n,q,?] (default: n)
```

Adding a Disk to a Disk Group

You may wish to add a new disk to an already established disk group. For example, the current disks may have insufficient space for the project or work group requirements, especially if these requirements have changed. You can add a disk to a disk group by following the steps required to add a disk. See "Bringing Physical Disks Under Volume Manager Control" on page 128.

Creating a Disk Group

You may want all data related to a particular set of applications of a particular group of users needs to be made accessible on another system. Examples of this are:

- A system has failed and its data must be moved to other systems.
- The work load must be balanced across a number of systems.

In these cases, it is important that the data related to particular application(s) or users be located on an identifiable set of disk drives. When these disks are moved, all data belonging to the application(s) or users on these disks, and no other data, is moved.

Note The Volume Manager supports a default disk group, `rootdg`, in which all volumes are created if no further specification is given. All commands default to `rootdg` as well.

A disk group can only be created along with a disk. A disk group must have at least one disk associated with it.

If you need to create a disk group in addition to `rootdg`, do the following:

1. Select menu item `1 (Add or initialize one or more disks)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the disk device name of the disk to be added to Volume Manager control:

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

Use this operation to add one or more disks to a disk group.
You can add the selected disks to an existing disk group or
to a new disk group that will be created as a part of the
operation. The selected disks may also be added to a disk
group as spares. The selected disks may also be initialized
without adding them to a disk group leaving the disks
available for use as replacement disks.

More than one disk or pattern may be entered at the prompt.
Here are some disk selection examples:

all:            all disks
c3 c4t2:        all disks on both controller 3 and controller
                4,target 2
c3t4d0:         a single disk

Select disk devices to add:
[<pattern-list>,all,list,q,?] c1t2d0
```

Where <*pattern-list*> can be a single disk, or a series of disks and/or controllers (with optional targets). If <*pattern-list*> consists of multiple items, those items must be separated by white space.

If you do not know the address (device name) of the disk to be added, enter **l** or **list** at the prompt for a listing of all disks.

3. To continue with the operation, enter **y** (or press Return) at the following prompt:

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

c1t2d0

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

4. At the following prompt, specify the disk group to which the disk should be added (anotherdg, in this case):

```
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)
anotherdg
```

5. vxdiskadm confirms that no active disk group currently exists with the same name and prompts for confirmation that you really want to create this new disk group:

```
There is no active disk group named anotherdg.

Create a new group named anotherdg? [y,n,q,?] (default: y) y
```

6. At the following prompt, either press Return to accept the default disk name or enter a disk name:

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

7. When vxdiskadm asks whether to exclude this disk from hot-relocation use, enter **n** (or press Return):

```
Exclude disk from hot-relocation use? [y,n,q,?] (default: n)
```

8. When vxdiskadm asks whether this disk should become a hot-relocation spare, enter **n**(or press Return):

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

9. To continue with the operation, enter **y** (or press Return) at the following prompt:

    ```
    A new disk group will be created named anotherdg and the
    selected disks will be added to the disk group with default disk
    names.

    c1t2d0

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

10. The following prompt appears if this disk was previously initialized for future Volume Manager use; enter **y** to confirm that you now want to use this disk:

    ```
    The following disk device appears to have been initialized
    already. The disk is currently available as a replacement disk.
    Output format: [Device_Name]

    c1t2d0

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

    If you are sure you want to reinitialize the disk, enter **y** (or press Return) at the following prompt:

    ```
    The following disk you selected for use appears to already have
    been initialized for the Volume Manager.  If you are certain the
    disk has already been initialized for the Volume Manager, then
    you do not need to reinitialize the disk device.
    Output format: [Device_Name]

    c1t2d0

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

    Messages similar to the following confirm that this disk is being reinitialized for Volume Manager use:

    ```
    Initializing device c1t2d0.

    Creating a new disk group named anotherdg containing the disk
    device c1t2d0 with the name another01.
    ```

11. At the following prompt, indicate whether you want to continue to initialize more disks (**y**) or return to the `vxdiskadm` main menu (**n**):

    ```
    Add or initialize other disks? [y,n,q,?] (default: n)
    ```

Deporting a Disk Group

Use the deport disk group task to disable access to a disk group that is currently enabled (imported) by this system. Deport a disk group if you intend to move the disks in a disk group to another system. Also, deport a disk group if you want to use all of the disks remaining in a disk group for some new purpose.

To deport a disk group, do the following:

1. Select menu item `9 (Remove access to (deport) a disk group)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the name of the disk group to be deported:

   ```
   Remove access to (deport) a disk group
   Menu: VolumeManager/Disk/DeportDiskGroup

   Use this menu operation to remove access to a disk group that is
   currently enabled (imported) by this system. Deport a disk
   group if you intend to move the disks in a disk group to another
   system. Also, deport a disk group if you want to use all of the
   disks remaining in a disk group for some new purpose.

   You will be prompted for the name of a disk group. You will also
   be asked if the disks should be disabled (offlined). For
   removable disk devices on some systems, it is important to
   disable all access to the disk before removing the disk.

   Enter name of disk group [<group>,list,q,?] (default: list)
   newdg
   ```

3. At the following prompt, enter **y** if you intend to remove the disks in this disk group:

   ```
   The requested operation is to disable access to the removable
   disk group named newdg. This disk group is stored on the
   following disks:
   newdg01 on device c1t1d0s2

   You can choose to disable access to (also known as “offline”)
   these disks. This may be necessary to prevent errors if
   you actually remove any of the disks from the system.

   Disable (offline) the indicated disks? [y,n,q,?] (default: n)
   ```

4. At the following prompt, press Return to continue with the operation:

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

   Once the disk group is deported, `vxdiskadm` displays the following message:

   ```
   Removal of disk group newdg was successful.
   ```

5. At the following prompt, indicate whether you want to disable another disk group (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Disable another disk group? [y,n,q,?] (default: n)
   ```

Importing a Disk Group

Use this menu task to enable access by this system to a disk group. If you wish to move a disk group from one system to another, you must first disable (deport) it on the original system, then move the disk between systems and enable (import) the disk group.

To import a disk group, do the following:

1. Select menu item `8 (Enable access to (import) a disk group)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the name of the disk group to import:

   ```
   Enable access to (import) a disk group
   Menu: VolumeManager/Disk/EnableDiskGroup

   Use this operation to enable access to a disk group. This can be
   used as the final part of moving a disk group from one system to
   another. The first part of moving a disk group is to use the
   “Remove access to (deport) a disk group” operation on the
   original host.

   A disk group can be imported from another host that failed
   without first deporting the disk group. Be sure that all disks
   in the disk group are moved between hosts.

   If two hosts share a SCSI bus, be very careful to ensure that
   the other host really has failed or has deported the disk group.
   If two active hosts import a disk group at the same time, the
   disk group will be corrupted and will become unusable.

   Select disk group to import [<group>,list,q,?] (default: list)
   newdg
   ```

   Once the import is complete, `vxdiskadm` displays the following success message:

   ```
   The import of newdg was successful.
   ```

3. At the following prompt, indicate whether you want to import another disk group (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Select another disk group? [y,n,q,?] (default: n)
   ```

Exiting vxdiskadm

When you have completed all of your disk administration activities, exit `vxdiskadm` by selecting **q** from the main menu.

Volume Tasks 5

Introduction

This chapter describes how to create and maintain a system configuration under Volume Manager control. It includes information about creating, removing, and maintaining Volume Manager objects (volumes, plexes, and subdisks). Online backup information is also included.

Volume, Plex, and Subdisk Tasks and Online Backup

This chapter provides information on the following tasks:

Volume Tasks

- Creating Volumes
- Resizing a Volume
- Removing a Volume
- Mirroring a Volume
- Removing a Mirror
- Adding a RAID-5 Log
- Adding a DRL Log
- Removing a DRL Log
- Removing a RAID-5 Log
- Stopping a Volume
- Starting a Volume
- Changing the Volume Read Policy
- Preparing a Volume to Restore From Backup
- Recovering a Volume
- Displaying Volume Information

Plex Tasks

- Creating Plexes
- Associating Plexes
- Dissociating and Removing Plexes
- Backup Using Mirroring
- Displaying Plex Information
- Changing Plex Attributes
- Changing Plex Status: Detaching and Attaching Plexes
- Moving Plexes
- Copying Plexes

Subdisk Tasks

- Creating Subdisks
- Removing Subdisks
- Displaying Subdisk Information
- Associating Subdisks
- Dissociating Subdisks
- Changing Subdisk Attributes
- Moving Subdisks
- Splitting Subdisks
- Joining Subdisks

Performing Online Backup

- Fast Mirror Resynchronization
- Mirroring Volumes on a VM Disk
- Moving Volumes from a VM Disk

Note Some Volume Manager commands require superuser or other appropriate privileges.

Creating Volumes

Volumes are created to take advantage of the Volume Manager concept of virtual disks. Once a volume exists, a file system can be placed on the volume to organize the disk space with files and directories. Also, applications such as databases can be used to organize data on volumes.

Disks and disk groups must be initialized and defined to the Volume Manager before volumes can be created.

You can create volumes using either a basic or advanced approach:

- Basic—The basic approach takes information about what you want to accomplish and then performs the necessary underlying tasks. This approach requires only minimal input from you, but also permits more detailed specifications. Basic operations are performed primarily through the `vxassist` command. `vxassist` creates required plexes and subdisks by using only the basic attributes of the desired volume as input. `vxassist` can also modify existing volumes. It automatically modifies any underlying or associated objects. `vxassist` uses default values for many volume attributes, unless you provide specific values.
- Advanced—The advanced approach consists of a number of commands that typically require you to specify detailed input. These commands use a “building block” approach that requires you to have a detailed knowledge of the underlying structure and components to manually perform the commands necessary to accomplish a certain task. Advanced operations are performed through several Volume Manager commands.

The steps normally involved in creating volumes manually are:

- create subdisks
- create plexes
- associate subdisks and plexes
- create volumes
- associate volumes and plexes
- initialize volumes

The creation of a volume involves the creation of plex and subdisk components. With the basic approach to volume creation, you indicate the desired volume characteristics and the underlying plexes and subdisks are created automatically.

Volumes can be created with various layout types:

- Concatenated—A volume whose subdisks are arranged both sequentially and contiguously within a plex. Concatenation allows a volume to be created from multiple regions of one or more disks if there is not enough space for an entire volume on a single region of a disk.

- Striped—A volume with data spread evenly across multiple disks. *Stripes* are equal-sized fragments that are allocated alternately and evenly to the subdisks of a single plex. There must be at least two subdisks in a striped plex, each of which must exist on a different disk. Throughput increases with the number of disks across which a plex is striped. Striping helps to balance I/O load in cases where high traffic areas exist on certain subdisks.
- RAID-5—A volume that uses striping to spread data and parity evenly across multiple disks in an array. Each stripe contains a parity stripe unit and data stripe units. Parity can be used to reconstruct data if one of the disks fails. In comparison to the performance of striped volumes, write throughput of RAID-5 volumes decreases since parity information needs to be updated each time data is accessed. However, in comparison to mirroring, the use of parity reduces the amount of space required.
- Mirrored—A volume with multiple plexes that duplicate the information contained in a volume. Although a volume can have a single plex, at least two are required for true mirroring (redundancy of data). Each of these plexes should contain disk space from different disks, for the redundancy to be useful.
- Layered Volume—A volume built on top of volumes. Layered volumes can be constructed by mapping a subdisk to a VM disk or to a *storage* volume. A storage volume provides a recursive level of layout that is similar to the top-level volumes. Layered volumes allow for more combinations of logical layouts.

Note In the VERITAS Volume Manager Storage Administrator, `Striped-Pro` is the GUI term used for a striped-mirror, and `Concatenated-Pro` is the GUI term used for a concatenated-mirror.

The `vxassist` command provides the simplest way to create new volumes. You can create a a volume with `vxassist` as follows:

```
# vxassist make volume_name length [attributes]
```

where `make` is the keyword for volume creation, *volume_name* is a name you give to the volume, and *length* specifies the number of sectors (by default) in the volume. The length can be specified in kilobytes, megabytes, or gigabytes by using a suffix character of `k`, `m`, or `g`, respectively. Refer to the `vxintro` (1M) manual page for more information on specifying units of length when creating volumes. Additional attributes can be specified, as appropriate.

By default, `vxassist` creates volumes in the `rootdg` disk group. Another disk group can be specified by including `-g` *diskgroup* in the `vxassist` command line.

Creating a volume in the disk group `rootdg` creates two device node files that can be used to access the volume:

- `/dev/vx/dsk/`*`volume_name`* (the block device node for the volume)
- `/dev/vx/rdsk/`*`volume_name`* (the raw device node for the volume)

For volumes in `rootdg` and disk groups other than `rootdg`, these names include the disk group name, as follows:

- `/dev/vx/dsk/`*`diskgroup_name/volume_name`*
- `/dev/vx/rdsk/`*`diskgroup_name/volume_name`*

"Creating a Concatenated Volume" on page 163 describes the simplest way to create a (default) volume. Later sections describe how to create volumes with specific attributes.

Creating a Concatenated Volume

By default, `vxassist` creates a concatenated volume that uses one or more sections of disk space. On a fragmented disk, this allows you to put together a volume larger than any individual section of free disk space available.

If there is not enough space on a single disk, `vxassist` creates a spanned volume. A spanned volume is a concatenated volume with sections of disk space spread across more than one disk. A spanned volume can be larger than the single largest disk, since it takes space from more than one disk.

Creating a Concatenated Volume on Any Disk

If no disk is specified, the Volume Manager selects a disk on which to create the volume.

Create a concatenated, default volume with this command:

```
# vxassist make volume_name length
```

where *volume_name* is the name of the volume and *length* specifies the length of the volume in sectors (unless another unit of size is specified with a suffix character). When you create a volume, you can specify the length of a new volume in sectors, kilobytes, megabytes, or gigabytes. The unit of measure is added as a suffix to the length (`s`, `m`, `k`, or `g`). If no unit is specified, sectors are assumed.

You can use this command to create the volume `voldefault` with a length of 10 megabytes:

```
# vxassist make voldefault 10m
```

Creating a Concatenated Volume on a Specific Disk

The Volume Manager automatically selects the disk(s) each volume resides on, unless you specify otherwise. If you want a volume to reside on a specific disk, you must designate that disk for the Volume Manager. More than one disk can be specified.

Create a volume on a specific disk with this command:

```
# vxassist make volume_name length diskname [...]
```

To create the volume volspecific on disk03, use this command:

```
# vxassist make volspecific 3m disk03
```

Creating a Striped Volume

A striped volume contains at least one plex that consists of two or more subdisks located on two or more physical disks.

Create a striped volume with this command:

```
# vxassist make volume_name length layout=stripe
```

To create the striped volume volzebra, use this command:

```
# vxassist make volzebra 10m layout=stripe
```

This creates a striped volume with the default stripe unit size on the default number of disks.

You can indicate the disks on which the volumes are to be created by specifying the disk names at the end of the command line. For example, to create a 30 megabyte striped volume on three specific disks (disk03, disk04, and disk05), use this command:

```
# vxassist make stripevol 30m layout=stripe disk03 disk04\disk05
```

Creating a RAID-5 Volume

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. A RAID-5 volume can also contain one or more RAID-5 log plexes, which are used to log information about data and parity being written to the volume.

Create a RAID-5 volume with this command:

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

To create the RAID-5 volume volraid, use this command:

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

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

Creating a Mirrored Volume

Create a new mirrored volume with this command:

```
# vxassist make volume_name length layout=mirror
```

To create the mirrored volume, volmir, use this command:

```
# vxassist make volmir 5m layout=mirror
```

Resizing a Volume

Resizing a volume changes the volume size. This can be done by using either vxassist or vxresize.

If the volume is not large enough for the amount of data that needs to be stored in it, you need to extend the length of the volume. If a volume is increased in size, vxassist automatically finds available disk space.

When you resize a volume, you can specify the length of a new volume in sectors, kilobytes, megabytes, or gigabytes. The unit of measure is added as a suffix to the length (s, m, k, or g). If no unit is specified, sectors are assumed.

Caution Do not shrink a volume below the size of the file system. If you have a VxFS file system, you can shrink the file system and then shrink the volume. If you do not shrink the file system first, you risk unrecoverable data loss.

Resizing Volumes With vxassist

vxassist can resize a volume in 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 You cannot grow or shrink any volume associated with an encapsulated bootdisk (rootvol, usr, var, opt, swapvol, etc.) because these map to a physical underlying partition on the disk and must be contiguous. If you attempt to grow rootvol, usrvol, varvol, or swapvol the system could become unbootable if you need to revert back to slices to boot. It can also prevent a succesful Solaris upgrade and you might have to do a fresh install. Additionally, the upgrade_start script might fail.

Extending to a Given Length

Extend a volume *to* a specific length with this command:

```
# vxassist growto volume_name length
```

To extend volcat to 2000 sectors, use this command:

```
# vxassist growto volcat 2000
```

Extending by a Given Length

Extend a volume *by* a specific length with this command:

```
# vxassist growby volume_name length
```

To extend volcat by 100 sectors, use this command:

```
# vxassist growby volcat 100
```

Shrinking to a Given Length

Shrink a volume *to* a specific length with this command:

```
# vxassist shrinkto volume_name length
```

Make sure you do not shrink the volume below the current size of the file system or database using the volume. This command can be safely used on empty volumes.

To shrink volcat to 1300 sectors, use this command:

```
# vxassist shrinkto volcat 1300
```

Shrinking by a Given Length

Shrink a volume *by* a specific length with this command:

```
# vxassist shrinkby volume_name length
```

To shrink volcat by 300 sectors, use this command:

```
# vxassist shrinkby volcat 300
```

Resizing Volumes With vxresize

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.

If your volume is larger than you need it to be, you can shrink the size of the volume.

Removing a Volume

Once a volume is no longer necessary (it is inactive and archived, for example), you can remove the volume and free up the disk space for other uses by using the command vxassist remove volume.

You can remove an entire volume with the vxassist command. Use the keywords remove and volume and provide the volume name on the command line as shown in this example:

```
# vxassist remove volume volume_name
```

Mirroring a Volume

A mirror is a copy of a volume that is not stored on the same disk(s) as the original copy of the volume. Mirroring a volume ensures that the data in that volume is not lost if one of your disks fails.

Creating a Mirrored Volume

To create a mirrored volume witrh vxassist:

```
# vxassist make vol_name  length layout=mirror
```

Mirroring an Existing Volume

A mirror (plex) can be added to an existing volume. This can be done with the vxassist command:

```
# vxassist mirror volume_name
```

For example:

```
# vxassist mirror voltest
```

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

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

Creating a Volume with Dirty Region Logging Enabled

To create a mirrored volume with Dirty Region Logging (DRL) enabled, create a mirrored volume with a log with this command:

```
# vxassist make volume_name length layout=mirror,log
```

vxassist creates one log plex for each log subdisk, by default.

Mirroring All Volumes

To mirror all existing volumes on the system to available disk space, use this command:

```
# /etc/vx/bin/vxmirror -g diskgroup -a
```

You can also configure the Volume Manager to create mirrored volumes by default. To do this, enter the command:

```
# /etc/vx/bin/vxmirror -d yes
```

If you make this change, you can still make unmirrored volumes by specifying nmirror=1 as an attribute to the vxassist command. For example, to create an unmirrored 20-megabyte volume named nomirror, use the command:

```
# vxassist make nomirror 20m nmirror=1
```

Removing a Mirror

When a mirror is no longer needed, you can remove it. A mirror may be removed to provide free disk space.

Note The last valid plex associated with a volume cannot be removed.

Caution To save the data on a mirror to be removed, the configuration of that mirror must be known. Parameters from that configuration (stripe unit size and subdisk ordering) are critical to the creation of a new mirror to contain the same data. Before this type of mirror is removed, its configuration must be recorded.

A mirror can be dissociated and removed from the associated volume with this command:

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

To dissociate and remove a mirror named vol01-02, use this command:

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

This removes the mirror vol01-02 and all associated subdisks.

Adding a RAID-5 Log

Only one RAID-5 plex can exist per RAID-5 volume. Any additional plexes are RAID-5 log plexes, which are used to log information about data and parity being written to the volume. When a RAID-5 volume is created using vxassist, a log plex is created for that volume by default.

Add a RAID-5 log to an existing volume with this command:

```
# vxassist addlog volume_name
```

To create a log for the RAID-5 volume `volraid`, use this command:

```
# vxassist addlog volraid
```

Adding a DRL Log

To put Dirty Region Logging into effect for a volume, a log subdisk must be added to that volume and the volume must be mirrored. Only one log subdisk can exist per plex.

Add a DRL log to an existing volume with this command:

```
# vxassist addlog volume_name
```

To create a log for the volume `vol03`, use this command:

```
# vxassist addlog vol03
```

When `vxassist` is used to add a log subdisk to a volume, a log plex is also created to contain the log subdisk by default.

Once created, the plex containing a log subdisk can be treated as a regular plex. Data subdisks can be added to the log plex. The log plex and log subdisk can be removed using the same procedures used to remove ordinary plexes and subdisks.

Removing a DRL Log

You can remove a log with the `vxassist` command as follows.

```
# vxassist remove log volume_name
```

Use the attribute `nlog=` to specify the number of logs to be removed. By default, `vxassist` removes one log.

Removing a RAID-5 Log

You can remove a RAID-5 log with the `vxassist` command as follows:

```
# vxassist remove log volume_name
```

Use the attribute `nlog=` to specify the number of logs to be removed. By default, `vxassist` removes one log.

Stopping a Volume

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

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

For example, the command to stop a volume named `vol01` is:

```
# vxvol stop vol01
```

To stop all ENABLED volumes, use this command:

```
# vxvol stopall
```

Starting a Volume

Starting a volume makes it available for use.

Starting a volume changes the volume state from DISABLED or DETACHED to ENABLED. If a volume cannot be enabled, it remains in its current state. To start a volume, use this command:

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

To start all DISABLED volumes, use this command:

```
# vxrecover -s
```

Listing Unstartable Volumes

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

```
# vxinfo [volume_name
```

Changing the Volume Read Policy

Volume Manager offers the choice of the following read policies:

- `round` reads each plex in turn in “round-robin” fashion for each nonsequential I/O detected. Sequential access causes only one plex to be accessed. This takes advantage of the drive or controller read-ahead caching policies.

- `prefer` reads first from a plex that has been named as the preferred plex.
- `select` chooses a default policy based on plex associations to the volume. If the volume has an enabled striped plex, `select` defaults to preferring that plex; otherwise, it defaults to round-robin.

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

To set the read policy to `round`, use this command:

```
# vxvol rdpol round volume_name
```

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

```
# vxvol rdpol round vol01
```

To set the read policy to `prefer`, use this command:

```
# vxvol rdpol prefer volume_name preferred_plex_name
```

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

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

To set the read policy to `select`, use this command:

```
# vxvol rdpol select volume_name
```

Preparing a Volume to Restore From Backup

It is important to make backup copies of your volumes. This provides a copy of the data as it stands at the time of the backup. Backup copies are used to restore volumes lost due to disk failure, or data destroyed due to human error. The Volume Manager allows you to back up volumes with minimal interruption to users.

You can back up a volume with `vxassist` by using the following procedure:

1. Create a snapshot mirror of the volume to be backed up.

   The `vxassist snapstart task` creates a write-only backup mirror, which is attached to and synchronized with the volume to be backed up. When synchronized with the volume, the backup mirror is ready to be used as a snapshot mirror. However, it continues being updated until it is detached during the actual snapshot portion of the procedure. This may take some time, depending on the volume size.

   Create a snapshot mirror for a volume with this command:

```
# vxassist snapstart volume_name
```

To create a snapshot mirror of a volume called `voldef`, use this command:

```
# vxassist snapstart voldef
```

2. Choose a suitable time to create a snapshot volume.

   If possible, plan to take the snapshot at a time when users are accessing the volume as little as possible.

3. Create a snapshot volume that reflects the original volume at the time of the snapshot.

   The online backup procedure is completed by running the `vxassist snapshot` command on the volume with the snapshot mirror. This task detaches the finished snapshot mirror, creates a new normal volume, and attaches the snapshot mirror to it. The snapshot then becomes a read-only volume. This step should only take a few minutes.

   Create a snapshot volume with this command:

   ```
   # vxassist snapshot volume_name new_volume_name
   ```

   To create a snapshot volume of `voldef`, use this command:

   ```
   # vxassist snapshot voldef snapvol
   ```

   The snapshot volume can now be used by backup utilities while the original volume continues to be available for applications and users.

You can back up the snapshot volume by whatever means you prefer. To avoid wasting space, you should remove the snapshot volume when your backup is complete, The snapshot volume occupies as much space as the original volume.

Recovering a Volume

A system crash or an I/O error can corrupt one or more plexes of a volume and leave no plex CLEAN or ACTIVE. You can 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 the CLEAN state, use this command:

```
# vxmend fix clean plex_name
```

For example, the command to place the plex named `vol01-02` in the CLEAN state is:

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

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

Displaying Volume Information

You can use the `vxprint` command to display information about how a volume is configured.

You can display the volume, plex, and subdisk record information for all volumes in the system with this command:

```
# vxprint -ht
```

This is an example of the `vxprint` output:

```
Disk group: rootdg

DG NAME       NCONFIG  NLOG    MINORS   GROUP-ID
DM NAME       DEVICE   TYPE    PRIVLEN  PUBLEN   STATE
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

dg rootdg  759096729.1025.tweety
dm disk10     c1t0d0s2 sliced  559      1044400  -
dm disk20     c2t0d0s2 sliced  559      1044400  -

v  pubs       fsgen    ENABLED ACTIVE   2288    SELECT     -
pl pubs-01     pubs    ENABLED ACTIVE   2288     CONCAT     -    RW
sd disk10-01 pubs-01   disk10  0        2288     0          c0t0d0 ENA

v  voldef     sgen     ENABLED ACTIVE   20480    SELECT     -
pl voldef-01 voldef    ENABLED ACTIVE   20480    CONCAT     -     RW
sd disk10-02 voldef-0  disk10  2288     20480    0          c0t1d0 ENA
```

where `dg` is a disk group, `dm` is a disk, `v` is a volume, `pl` is a plex, and `sd` is a subdisk. The top few lines indicate the headers that match each type of output line that follows. Each volume is listed along with its associated plex(es) and subdisk(s).

You can display volume-related information for a specific volume with this command:

```
# vxprint -t  volume_name
```

To display information about `voldef`, use this command:

```
# vxprint -t voldef
```

This is an example of the `vxprint` output:

```
Disk group: rootdg

V  NAME    USETYPE  KSTATE   STATE    LENGTH  READPOL  PREFPLEX

v  voldef  fsgen    ENABLED  ACTIVE   20480   SELECT   -
```

Plex Tasks

Plexes are logical groupings of subdisks that create an area of disk space independent of physical disk size or other restrictions. Replication (mirroring) of disk data is done by creating multiple plexes for a single volume. Each plex contains an identical copy of the volume data. Because 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 tasks include:

- creating a plex
- backup 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

Creating Plexes

The `vxmake` command creates Volume Manager objects, such as plexes. When you create a plex, you identify subdisks and associate them to the plex that you want to create.

To create a plex from existing subdisks, use this command:

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

For example, the command to create a concatenated plex named `vol01-02` using two existing subdisks named `disk02-01` and `disk02-02` is:

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

Creating a Striped Plex

To create a striped plex, you need to specify additional attributes. For example, the command to create a striped plex named `pl-01` with a stripe width of 32 sectors and 2 columns is:

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

If you intend to use a plex to build a volume, you must associate the plex with the volume (see "Associating Plexes" on page 175 for more information.

Associating Plexes

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

```
# vxplex att volume_name plex_name
```

For example, the command to associate a plex named `vol01-02` with a volume named `vol01` is:

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

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

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

For example, the command to create a mirrored, `fsgen`-type volume named `home` and associate two existing plexes named `home-1` and `home-2` is:

```
# 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`*

Dissociating and Removing Plexes

When a plex is no longer needed, you can remove it. You may need to remove a plex in the following situations:

- to provide free disk space
- to reduce the number of mirrors in a volume so you can increase the length of another mirror and its associated volume. When the plexes and subdisks are removed, the resulting space can be added to other volumes
- to remove a temporary mirror that was created to back up a volume and is no longer needed
- to change the layout of a plex

Caution To save the data on a plex to be removed, the configuration of that plex must be known. Parameters from that configuration (stripe unit size and subdisk ordering) are critical to the creation of a new plex to contain the same data. Before a plex is removed, you must record its configuration. See "Displaying Plex Information" on page 177 for more information.

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

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

To dissociate and remove a plex named `vol01-02`, use this command:

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

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

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 as `vxplex -o rm dis`.

Backup Using Mirroring

If a volume is mirrored, backup can be done on that volume by taking one of the volume mirrors offline for a period of time. This removes the need for extra disk space for the purpose of backup only. However, it also removes redundancy of the volume for the duration of the time needed for the backup to take place.

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

You can perform backup of a mirrored volume on an active system with these steps:

1. Optionally stop user activity for a short time to improve the consistency of the backup.

2. Dissociate one of the volume mirrors (`vol01-01`, for this example):

   ```
   # vxplex dis vol01-01
   ```

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

   ```
   # vxmake -U gen vol tempvol plex=vol01-01
   ```

4. Start the temporary volume:

   ```
   # vxvol start tempvol
   ```

5. Do 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 tempvol
   ```

For information on an alternative online backup method using the vxassist command, refer to “Performing Online Backup.”

Displaying Plex Information

Listing plexes helps identify free plexes for building volumes. Using the vxprint command with the plex (–p) option lists information about all plexes.

To display detailed information about all plexes in the system, use this command:

```
# vxprint -lp
```

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

```
# vxprint -l plex_name
```

The –t option prints a single line of information about the plex. To list free plexes, use this command:

```
# vxprint -pt
```

Changing Plex Attributes

Caution Change plex attributes with extreme care, and only if necessary.

The vxedit command changes the attributes of plexes and other volume Manager objects. To change plex attributes, use this command:

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

The comment field and the putil and tutil fields are used by Volume Manager commands 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 functions and are numbered according to those functions. These fields can be modified as needed. Volume Manager uses the utility fields marked putil0 and tutil0. Other VERITAS products use those marked putil1 and tutil1. Fields marked putil2 and tutil2 are user fields. Table 4 on page 178 lists the functions of the putil and tutil fields.

Table 4. The putil[n] and tutil[n] Fields

Field	Description of Utility Fields
`putil0`	Reserved for use by Volume Manager commands and is retained on reboot.
`putil1`	Reserved for use by high-level utilities such as the graphical user 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 Volume Manager commands and is cleared on reboot.
`tutil1`	Reserved for use by high-level utilities such as the graphical user 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.

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 this command:

```
# vxedit set putil0=”DO-NOT-USE” vol01-02
```

Changing Plex Status: Detaching and Attaching Plexes

Once a volume has been created and placed online (ENABLED), Volume Manager can temporarily disconnect plexes 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 subdisks of that plex, use this command:

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

If a disk has a head crash, you should put all plexes that have associated subdisks on the affected disk OFFLINE. For example, if plexes `vol01-02` and `vol02-02` had subdisks on a drive to be repaired, use this command:

```
# 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 changed.

Detaching Plexes

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

```
# vxplex det plex_name
```

For example, the command to temporarily detach a plex named `vol01-02` and place it in maintenance mode is:

```
# vxplex det vol01-02
```

This command temporarily detaches the plex, but maintains the association between the plex and its volume. However, the plex is not used for I/O. A plex detached with the preceding command is recovered at 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 is recovered and made ACTIVE.

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

- If the volume is not ENABLED, start it with this command:

  ```
  # vxvol start volume_name
  ```

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

  ```
  # vxmend fix clean plex_name
  ```

  and then start the volume.

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

  ```
  # 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.

Attaching Plexes

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

If the volume is currently ENABLED, use this command:

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

For example, the command for a plex named `vol01-02` on a volume named `vol01` is:

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

This command 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), use this command:

```
# vxmend on plex_name
```

For example, the command for a plex named `vol01-02` is:

```
# vxmend on vol01-02
```

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

If it becomes necessary to manually change the state of a plex, refer to “Recovering a Volume.” in “Volume Tasks.” See the `vxmake`(1M) and `vxmend`(1M) manual pages for more information about these commands.

Moving Plexes

Moving a plex copies the data content from the original plex onto a new plex. The command to move data from one plex to another is:

```
# vxplex mv original_plex new_plex
```

For a move task 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 implications:

- If the new plex is smaller or more sparse than the original plex, an incomplete copy of the data on the original plex results. If this is the desired action, you must 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 is copied onto the new plex. Any area that was not on the original plex, but is represented on the new plex, is filled from other complete plex(es) 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 is not initialized and remains unused.

Copying Plexes

This task copies the contents of a volume onto a specified plex. The volume to be copied cannot be enabled. The plex cannot be associated with any other volume. To copy a plex, use this command:

```
# vxplex cp volume_name new_plex
```

After the copy task is complete, *new_plex* is not 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 If the plex being copied is larger than the volume, an incomplete copy of the data results. For the same reason, *new_plex* should not be sparse.

Subdisk Tasks

Subdisks are low-level building blocks in a Volume Manager configuration and used to build volumes. The following sections describe tasks that you can perform on subdisks.

Creating Subdisks

Note When you use vxassist or Storage Administrator to create volumes, the subdisks are automatically created for you.

You can use the `vxmake` command to create Volume Manager objects, such as subdisks. When you create a subdisk, you must specify the name and length of the subdisk as well as the starting point (offset) of the subdisk within the disk and the disk media name.

To create a subdisk, use this command:

```
# vxmake sd subdisk_name disk,offset,len
```

For example, the command to create a subdisk named `disk02-01` that starts at the beginning of disk `disk02` and has a length of 8000 sectors is shown here:

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

By default, Volume Manager commands take sizes in sectors. Adding a suffix (such as `k`, `m`, or `g`) changes the unit of size.

If you intend to use the new subdisk to build a volume, you must associate the subdisk with a plex (see“Associating Subdisks” on page 182). Subdisks for all plex layouts (concatenated, striped, RAID-5) are created the same way.

Removing Subdisks

To remove a subdisk, use this command:

```
# vxedit rm subdisk_name
```

For example, you can use this command to remove a subdisk named `disk02-01`:

```
# vxedit rm disk02-01
```

Displaying Subdisk Information

The `vxprint` command displays information about Volume Manager objects. To display general information for all subdisks, use this command:

```
# 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.

You can display complete information about a particular subdisk by using this command:

```
# vxprint -l subdisk_name
```

For example, here is the command to obtain all information on a subdisk named `disk02-01`:

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

This command provides the following output:

```
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/vx/dmp/c2t0d1s4 diskdev=32/68
```

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. The entire area that the subdisk fills must not be occupied by any portion of another subdisk. There are several ways that subdisks can be associated with plexes, depending on the overall state of the configuration.

If you have already created all the subdisks needed for a particular plex, associate subdisks at plex creation by using the command:

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

For example, the following command creates the plex `home-1` and associates subdisks `disk02-01`, `disk02-00`, and `disk02-02` with plex `home-1`.

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

Subdisks are associated in the listed order starting at offset 0. The disk space defined as `disk02-01` is first, the disk space `disk02-00` is second, and `disk02-02` is third. If you use this type of command, you do not have to specify the multiple commands needed to create the plex and then associate each of the subdisks with that plex. This method of associating subdisks is convenient during initial configuration.

Subdisks can also be associated with a plex that already exists. One or more subdisks can be associated with an existing plex with this command:

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

For example, you can use this command to associate subdisks named `disk02-01`, `disk02-00`, and `disk02-02` with a plex named `home-1`:

```
# 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 associates subdisks at a specific offset within the plex.

The `-l` option is needed when you have created a sparse plex (that is, a plex with a gap between its subdisks) for a particular volume, and want to make this plex complete. To make the plex complete, you need to create a subdisk of a size that fits the hole in the sparse plex exactly. Then you associate the subdisk with the plex by specifying the offset of the beginning of the hole in the plex. Use this command:

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

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

For striped subdisks, you can specify a column number and column offset for the subdisk:

```
# 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.

Associating Log Subdisks

Log subdisks are 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 "Dirty Region Logging." 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, care should be taken to position it on a disk that is not heavily used. Placing a log subdisk on a heavily-used disk can degrade system performance.

You can 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.

To add a log subdisk to an existing plex, use this command:

```
# vxsd aslog plex  subdisk
```

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

For example, you can use this command to associate a subdisk named `disk02-01` with a plex named `vol01-02` (which is already associated with volume `vol01`):

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

Dissociating Subdisks

To break an established connection 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 removed or used in another plex. To dissociate a subdisk, use this command:

```
# vxsd dis subdisk_name
```

For example, you can dissociate a subdisk named `disk02-01` from the plex with which it is currently associated, by using this command:

```
# vxsd dis disk02-01
```

Note You can also remove subdisks with the command:

```
# vxsd -orm dis subdisk_name
```

Changing Subdisk Attributes

Caution Change subdisk attributes with extreme care, and only if necessary.

The vxedit command changes attributes of subdisks to other Volume Manager objects. To change information relating to a subdisk, use this command:

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

For example, here is the command to change the comment field of a subdisk named disk02-01:

```
# vxedit set comment="new_comment" disk02-01
```

The subdisk fields that can be changed using vxedit are:

- name
- putil[*n*] fields
- 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 part of a plex.

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. The association is at the same offset within the plex as the source subdisk. To move a subdisk, use this command:

```
# vxsd mv old_subdisk_name new_subdisk_name
```

For the subdisk move task to work correctly, these 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.

Splitting Subdisks

Splitting a subdisk divides an existing subdisk into two subdisks. To split a subdisk, use this command:

```
# 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 to be created
- *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 to be created. The second subdisk occupies the remaining space used by the original subdisk.

If the original subdisk is associated with a plex before the task, upon completion of the split, both of the resulting subdisks are 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.

Joining Subdisks

Joining subdisks 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 subdisks, use this command:

```
# vxsd join subdisk1 subdisk2 new_subdisk
```

Performing Online Backup

Volume Manager provides snapshot backups of volume devices. This is done through `vxassist` and other commands. There are various procedures for doing backups, depending upon the requirements for integrity of the volume contents. These procedures have the same starting requirement: a plex 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 is by using the `vxassist` command which is easy to use. The `vxassist snapstart, snapwait`, and `snapshot tasks` provide a way to do online backup of volumes with minimal disruption to users.

The `vxassist snapshot` procedure consists of two steps:

1. Running `vxassist snapstart` to create a snapshot mirror

2. Running `vxassist snapshot` to create a snapshot volume

Note You can use the `vxassist` command to create a `snapshot` of a RAID-5 volume by using the recommended approach to volume backup described in this section.

The `vxassist snapstart` step 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 indicated by the new `snapshot` mirror changing its state to SNAPDONE. This change can be tracked by the `vxassist snapwait task`, 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. You can then select a convenient time at which to create a `snapshot` volume as an image of the existing volume. You 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 in contrast to the brief amount of time that it takes to create the `snapshot` volume.

The online backup procedure is completed by running the `vxassist snapshot` command on a volume with a SNAPDONE mirror. This task detaches the finished `snapshot` (which becomes a normal mirror), creates a new normal volume and attaches the `snapshot` mirror to the `snapshot` volume. 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:

1. Create a `snapshot` mirror for a volume with this command:

   ```
   # vxassist snapstart volume_name
   ```

2. When the `snapstart` step is complete and the mirror is in a SNAPDONE state, choose a convenient time to complete the `snapshot` task. 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 with this command:

   ```
   # vxassist snapshot volume_name temp_volume_name
   ```

4. Use `fsck` (or some utility appropriate for the application running on the volume) to clean the temporary volume's contents. For example, you can use this command:

   ```
   # 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 with this command:

   ```
   # vxedit -rf rm temp_volume_name
   ```

Fast Mirror Resynchronization

The Fast Mirror Resynchronization (FMR) feature performs quick and efficient resynchronization of stale mirrors by increasing the efficiency of the VxVM snapshot mechanism to better support operations such as backup and decision support.

Enabling FMR

When a new volume is created with `vxassist`, an attribute can be specified to turn FMR `on` or `off`. Both keywords `fmr` and `fastresync` can be used as attributes to specify that FMR will be used (or not) on a volume.

To create a volume with FMR enabled, use the `vxassist make` command as follows:

```
# vxassist make volume_name size fmr=on
```

The default is for FMR to be `off`, but you can change the default in the `vxassist` default file.

The FMR functionality can also be turned *ON* or *OFF* with the `vxvol` command. To use FMR, FMR must be enabled when the snapshot is taken, and FMR must remain enabled until after the snapback is completed.Turning FMR off will free all of the tracking maps for the specified volume. All subsequent reattaches will not use the FMR facility, but do a full resync of the volume. This occurs even if FMR is later turned on.

To turn FMR on, use the following command:

```
# vxvol set fmr=on volume_name
```

To turn FMR off, use the following command:

```
# vxvol set fmr=off volume_name
```

Merging a Snapshot Volume

A snapshot copy of a volume can be merged back with the original volume. The snapshot plex is detached from the snapshot volume and attached to the original volume. The snapshot volume is removed. This task resynchronizes the data in the volume so that the plexes are consistent.

To merge a snapshot with its original volume, use this command:

```
# vxassist snapback replica-volume
```

where *replica-volume* is the snapshot copy of the volume.

By default, the data in the original plex is used for the merged volume. To use the data copy from the replica volume instead, use this command:

```
# vxassist -o resyncfromreplica snapback replica-volume
```

Dissociating a Snapshot Volume

The link between a snapshot and its original volume can be permanently broken so that the snapshot volume becomes an independent volume.

To dissociate a snapshot from its original volume, use this command:

```
# vxassist snapclear replica-volume
```

where *replica-volume* is the snapshot copy of the volume.

Displaying Snapshot Volume Information

The `vxassist snapprint` command displays the associations between the original volumes and their respective replicas (snapshot copies).

The syntax for the `snapprint` option is:

```
# vxassist snapprint [volume-name]
```

The output from this command displays the following:

```
V NAME  USETYPE  LENGTH  RP NAME    VOLUME     LENGTH  RRPLEXID
v vol   fsgen    2048    rp vol-05  SNAP1-vol  3040    rp vol-04
```

If a volume is specified it will display either output for that volume or an error message if no FMR maps are enabled for that volume. Otherwise, it will display information for all volumes in the disk group.

Mirroring Volumes on a VM Disk

Mirroring the volumes on a VM disk gives you one or more copies of your volumes in another disk location. By creating mirror copies of your volumes, you protect your system against loss of data in case of a disk failure. You can use this task on your root disk to make a second copy of the boot information available on an alternate disk. This allows you to boot your system even if your root disk is corrupted.

Note This task only mirrors concatenated volumes. Volumes that are already mirrored or that contain subdisks that reside on multiple disks are ignored.

To mirror volumes on a disk, make sure that the target disk has an equal or greater amount of space as the originating disk and then do the following:

1. Select menu item `6 (Mirror volumes on a disk)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the disk name of the disk that you wish to mirror:

```
Mirror volumes on a disk
Menu: VolumeManager/Disk/Mirror

This operation can be used to mirror volumes on a disk. These
volumes can be mirrored onto another disk or onto any
available disk space. Volumes will not be mirrored if they are
already mirrored. Also, volumes that are comprised of more than
one subdisk will not be mirrored.

Mirroring volumes from the boot disk will produce a disk that
can be used as an alternate boot disk.

Enter disk name [<disk>,list,q,?] disk02
```

3. At the following prompt, enter the target disk name (this disk must be the same size or larger than the originating disk):

```
You can choose to mirror volumes from disk disk02 onto any
available disk space, or you can choose to mirror onto a
specific disk. To mirror to a specific disk, select the name of
that disk. To mirror to any available disk space, select "any".

Enter destination disk [<disk>,list,q,?] (default: any) disk01
```

Note Be sure to always specify the destination disk when you are creating an alternate root disk. Otherwise, the Volume Manager selects a disk to be the alternate root disk. However, your system may not be able to boot from that disk.

4. At the following prompt, press Return to make the mirror:

   ```
   The requested operation is to mirror all volumes on disk disk02
   in disk group rootdg onto available disk space on disk disk01.

   NOTE: This operation can take a long time to complete.

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

```
vxdiskadm displays the status of the mirroring operation:
     Mirror volume voltest-bk00 ...

     Mirroring of disk disk01 is complete.
```

5. At the following prompt, indicate whether you want to mirror volumes on another disk (**y**) or return to the `vxdiskadm` main menu (**n**):

   ```
   Mirror volumes on another disk? [y,n,q,?] (default: n)
   ```

Moving Volumes from a VM Disk

Before you disable or remove a disk, you may want to move the data from that disk to other disks on the system. To do this, make sure that the target disks have sufficient space, then do the following:

1. Select menu item `7 (Move volumes from a disk)` from the `vxdiskadm` main menu.

2. At the following prompt, enter the disk name of the disk whose volumes you wish to move:

   ```
   Move volumes from a disk
   Menu: VolumeManager/Disk/Evacuate

   Use this menu operation to move any volumes that are using a
   disk onto other disks. Use this menu immediately prior to
   removing a disk, either permanently or for replacement. You can
   specify a list of disks to move volumes onto, or you can move
   the volumes to any available disk space in the same disk group.

   NOTE: Simply moving volumes off of a disk, without also removing
   ```

```
            the disk, does not prevent volumes from being moved onto
        the disk by future operations. For example, using two
        consecutive move operations may move volumes from the
        second disk to the first.

      Enter disk name [<disk>,list,q,?] disk01
```

After the following display, you can optionally specify a list of disks to which the volume(s) should be moved.

```
      You can now specify a list of disks to move onto.  Specify a
list
      of disk media names (e.g., disk01) all on one line separated by
      blanks.  If you do not enter any disk media names, then the
      volumes will be moved to any available space in the disk group.
```

At the following prompt, press **Return** to move the volumes:

```
      Requested operation is to move all volumes from disk disk01 in
      group rootdg.

      NOTE: This operation can take a long time to complete.

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

```
As the volumes are moved from the disk, vxdiskadm displays the status
of the operation:
      Move volume voltest ...
      Move volume voltest-bk00 ...

When the volumes have all been moved, vxdiskadm displays the following
success message:
      Evacuation of disk disk01 is complete.
```

3. At the following prompt, indicate whether you want to move volumes from another disk (`y`) or return to the `vxdiskadm` main menu (`n`):

```
      Move volumes from another disk? [y,n,q,?] (default: n)
```

Volume Manager Cluster Functionality 6

Introduction

This chapter discusses the cluster functionality provided with the VERITAS Volume Manager (VxVM). The Volume Manager includes an optional cluster feature that enables VxVM to be used in a cluster environment. The cluster functionality in the Volume Manager is a separately licensable feature.

The following topics are covered in this chapter:

- Cluster Functionality Overview
- Disks in VxVM Clusters
- Dirty Region Logging and Cluster Environments
- Upgrading Volume Manager Cluster Functionality
- Cluster-related Volume Manager Utilities and Daemons
- Clustering and FastResync

For information about cluster-related error messages, refer to the error message chapter of the *VERITAS Volume Manager Reference Guide*.

Cluster Functionality Overview

The cluster functionality in the Volume Manager allows multiple hosts to simultaneously access and manage a given set of disks under Volume Manager control (*VM disks*). A *cluster* is a set of hosts sharing a set of disks; each host is referred to as a *node* in the cluster. The nodes are connected across a network. If one node fails, the other node(s) can still access the disks. The Volume Manager cluster feature presents the same logical view of the disk configurations (including changes) on all nodes.

Note With cluster support enabled, the Volume Manager supports up to four nodes per cluster.

The sections that follow provide more information on the cluster functionality provided by the Volume Manager.

Shared Volume Manager Objects

When the cluster feature is enabled, Volume Manager objects are capable of being shared by all of the nodes in a given cluster.

The Volume Manager cluster feature allows for two types of disk groups:

- *Private disk groups*, which belong to only one node. A private disk group is only imported by one system. Disks in a private disk group may be physically accessible from one or more systems, but actual access is restricted to one system only.
- *Cluster-shareable disk groups*, which are shared by all nodes. A cluster-shareable (or *shared*) disk group is imported by all cluster nodes. Disks in a cluster-shareable disk group must be physically accessible from all systems that may join the cluster.

In a Volume Manager cluster, most disk groups are shared. However, the root disk group (`rootdg`) is always a private disk group.

Disks in a shared disk group are accessible from all nodes in a cluster, allowing applications on multiple cluster nodes to simultaneously access the same disk. A volume in a shared disk group can be simultaneously accessed by more than one node in the cluster, subject to licensing and disk group activation mode descriptions.

A shared disk group must be activated on a node in order for the volumes in the disk group to become accessible for application I/O from that node. The ability of applications to read or write to volumes is dictated by the activation mode of the disk group. Valid activation modes for a shared disk group are *exclusive-write*, *shared-write*, *read-only*, *shared-read* and *off* (or inactive), as shown in Table 5, ”Activation Modes for Shared Disk Group.”

Table 5. Activation Modes for Shared Disk Group

Exclusive write	The node has exclusive write access to the disk group. No other node can activate the dg for write access.
Shared write	The node has write access to the disk group.
Read only	The node has read access to the disk group and denies write access for all other nodes in the cluster. The node has no write access to the disk group. Attempts to activate a disk group for either of the write modes on other nodes will fail.
Shared read	The node has read access to the disk group. The node has no write access to the disk group, however other nodes can obtain write access.
Off	The node has neither read nor write access to the disk group. Query operations on the disk group are permitted.

Note Disk group activation was a new feature in VxVM 3.0. To maintain compatibility with previous releases, activation modes are, by default, transparent to Volume Manager utilities. Shared disk groups are automatically activated in shared-write mode.

Special uses of clusters, such as HA applications and off-host backup, can utilize disk group activation to explicitly control volume I/O capability from different nodes in the cluster. Use of activation modes is described in “Disk Group Activation” on page 201.

Notes:

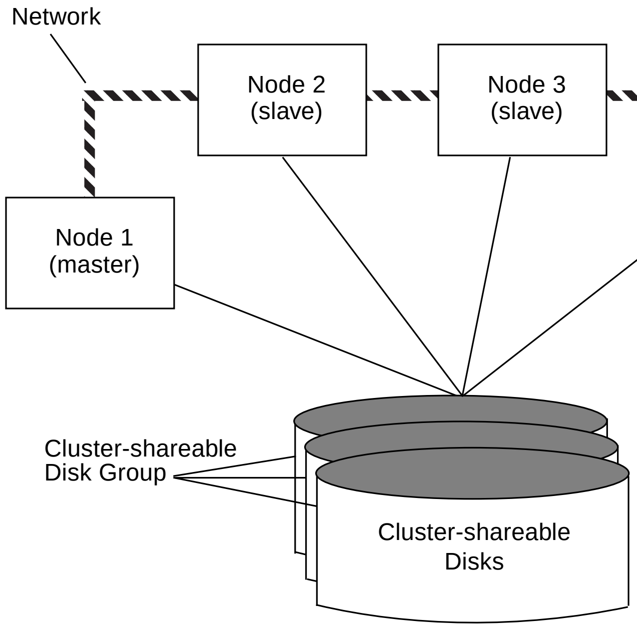
- The new features introduced in Volume Manager 3.0: *striped mirror volumes*, *task monitor*, and *online relayout*, are available in private disk groups, but are not yet supported for shared disk groups.
- Only raw device access is performed via the Volume Manager cluster feature. Shared volumes with file systems are not supported.
- The Volume Manager cluster feature does not currently support RAID-5 volumes in cluster-shareable disk groups. RAID-5 volumes can, however, be used in private disk groups attached to specific nodes of a cluster.
- If a desk group that contains unsupported objects is imported as shared, deport the disk group. Reorganize the contained volumes into supported layouts, and then reimport it as shared.

How Cluster Volume Management Works

The Volume Manager cluster feature works together with an externally-provided *cluster manager*, which is a daemon that informs VxVM of changes in cluster membership. Each node starts up independently and has its own copies of the operating system, VxVM with cluster support, and the cluster manager. When a node *joins* a cluster, it gains access to shared disks. When a node *leaves* a cluster, it no longer has access to those shared disks. The system administrator joins a node to a cluster by starting the cluster manager on that node.

Figure 24, ”Example of a 4-Node Cluster,” on page 196 illustrates a simple cluster arrangement. All of the nodes are connected by a network. The nodes are then connected to a cluster-shareable disk group. To the cluster manager, all nodes are the same. However, the Volume Manager cluster feature requires that one node act as the *master node*; the other nodes are *slave nodes*. The master node is responsible for coordinating certain Volume Manager activities. VxVM software determines which node performs the master function (any node is capable of being a master node); this role only changes if the master node leaves the cluster. If the master leaves the cluster, one of the slave nodes becomes the new master. In Figure 24, ”Example of a 4-Node Cluster,” Node 1 is the master node and Node 2, Node 3, and Node 4 are the slave nodes.

Figure 24. Example of a 4-Node Cluster



The system administrator designates a disk group as cluster-shareable using the `vxdg` utility (see "vxdg" on page 209 for more information). Once a disk group is imported as cluster-shareable for one node, the disk headers are marked with the cluster ID. When other nodes join the cluster, they will recognize the disk group as being cluster-shareable and import it. The system administrator can import or deport a shared disk group at any time; the operation takes places in a distributed fashion on all nodes.

Each physical disk is marked with a unique disk ID. When the cluster starts up on the master, it imports all the shared disk groups (except for any that have the `noautoimport` attribute set). When a slave tries to join, the master sends it a list of the disk IDs it has imported and the slave checks to see if it can access all of them. If the slave cannot access one of the imported disks on the list, it abandons its attempt to join the cluster. If it can access all of the disks on the list, it imports the same set of shared disk groups as the master and joins the cluster. When a node leaves the cluster, it deports all its imported shared disk groups, but they remain imported on the surviving nodes.

Any reconfiguration to a shared disk group is performed with the cooperation of all nodes. Configuration changes to the disk group happen simultaneously on all nodes and the changes are identical. These changes are atomic in nature, so they either occur simultaneously on all nodes or do not occur at all.

All members of the cluster have simultaneous read and write access to any cluster-shareable disk group depending on the activation mode. (See "Disk Group Activation" on page 201.) Access by the active nodes of the cluster is not affected by a failure in any other node. The data contained in a cluster-shareable disk group is available as long as at least one node is active in the cluster. Regardless of which node accesses the cluster-shareable disk group, the configuration of the disk group looks the same. Applications running on each node can access the data on the VM disks simultaneously.

Note VxVM does not protect against simultaneous writes to shared volumes by more than one node. It is assumed that any consistency control is done at the application level (using a distributed lock manager, for example).

Configuration & Initialization

Before any nodes can join a new cluster for the first time, the system administrator must supply certain configuration information. This information is supplied during cluster manager setup and is normally stored in some type of cluster manager configuration database. The precise content and format of this information is dependent on the characteristics of the cluster manager. The type of information required by VxVM is as follows:

- cluster ID
- node IDs
- network addresses of nodes
- port addresses

When a node joins the cluster, this information is automatically loaded into VxVM on that node at node startup time.

Node initialization is effected through the cluster manager startup procedure, which brings up the various cluster components (such as VxVM with cluster support, the cluster manager, and a distributed lock manager) on the node. Once it is complete, applications may be started. The system administrator invokes the cluster manager startup procedure on each node to be joined to the cluster.

For VxVM in a cluster environment, initialization consists of loading the cluster configuration information and joining the nodes in the cluster. The first node to join becomes the master node, and later nodes (slaves) join to the master. If two nodes join simultaneously, VxVM software chooses the master. Once the join for a given node is complete, that node has access to the shared disks.

Cluster Reconfiguration

Anytime there is a change in the state of the cluster (in the form of a node leaving or joining), a cluster reconfiguration occurs. Each node's cluster manager monitors other nodes in the cluster and informs VxVM when there is a change in cluster membership. The VxVM then takes appropriate action.

During a cluster reconfiguration, I/O to shared disks is suspended. It is resumed when the reconfiguration completes. Applications may therefore appear to be frozen for a short time.

If other operations (such as Volume Manager operations or recoveries) are in progress, the cluster reconfiguration may be delayed until those operations have completed. Volume reconfigurations (described later) do not take place at the same time as cluster reconfigurations. Depending on the circumstances, an operation may be held up and restarted later. In most cases, cluster reconfiguration takes precedence. However, if the volume reconfiguration is in the commit stage, it will complete first.

In case of SunCluster™ as a cluster manager, 'vxclust' utility coordinates cluster reconfigurations and provides communication between VxVM and SunCluster™. SunCluster™ and `vxclust` work together to ensure that each step in the cluster reconfiguration is completed in correct order.

In case of VCS as a clustering framework, the cluster reconfigurations are entirely handled inside VxVM kernel.

For more information on cluster reconfiguration,see "`vxclust`" on page 210 and "vxclustadm" on page 207.

Volume Reconfiguration

Volume reconfiguration is the process of creating, changing, and removing the Volume Manager objects in the configuration (such as disk groups, volumes, mirrors, etc.). In a cluster, this process is performed with the cooperation of all nodes. Volume reconfiguration is distributed to all nodes; identical configuration changes occur on all nodes simultaneously.

Note Volume reconfiguration is initiated and coordinated by the master node, so the system administrator must run the utilities that request changes to Volume Manager objects on the master node.

The vxconfigd daemons play an active role in volume reconfiguration`.`For the reconfiguration to succeed, `vxconfigd` must be running on all nodes.

The utility on the master node contacts its local `vxconfigd` daemon, which performs some local checking to make sure that a requested change is reasonable. For instance, it will fail an attempt to create a new disk group when one with the same name already exists. `vxconfigd` on the master node then sends messages with the details of the changes to the vxconfigd daemons on all other nodes in the cluster. The vxconfigds on each of the slave nodes then perform their own checking. For example, a slave node checks that it does not have a private disk group with the same name as the one being created; if the operation involves a new disk, each node checks that it can access that disk. When all of the vxconfigds on all nodes agree that the proposed change is reasonable, each vxconfigd notifies its kernel and the kernels then cooperate to either commit or abort the transaction. Before the transaction can commit, all of the kernels ensure that no I/O is underway. The master is responsible for initiating a reconfiguration and coordinating the transaction commit.

If vxconfigd on any node goes away during a reconfiguration process, all nodes will be notified and the operation will fail. If any node leaves the cluster, the operation will fail unless the master has already committed it. If the master leaves the cluster, the new master (which was a slave previously) either completes or fails the operation. This depends on whether or not it received notification of successful completion from the previous master. This notification is done in such a way that if the new master did not receive it, neither did any other slave.

If a node attempts to join the cluster while a volume reconfiguration is being performed, the results depend on how far the reconfiguration has got. If the kernel is not yet involved, the volume reconfiguration is suspended and restarted when the join is complete. If the kernel is involved, the join waits until the reconfiguration is complete.

When an error occurs (such as when a check on a slave fails or a node leaves the cluster), the error is returned to the utility and a message is issued to the console on the master node to identify the node on which the error occurred.

Node Shutdown

The system administrator can shut down the cluster on a given node by invoking the cluster manager’s shutdown procedure on that node. This terminates cluster components after cluster applications have been stopped. VxVM supports *clean node shutdown*, which is the ability of a node to leave the cluster gracefully when all access to shared volumes has ceased. The host is still operational, but cluster applications cannot be run on it.

The Volume Manager cluster feature maintains global state information for each volume. This enables VxVM to accurately determine which volumes need recovery when a node crashes. When a node leaves the cluster due to a crash or by some other means that is not clean, VxVM determines which volumes may have writes that have not completed and the master resynchronizes those volumes. If Dirty Region Logging is active for any of those volumes, it will be used.

Clean node shutdown should be used after, or in conjunction with, a procedure to halt all cluster applications. Depending on the characteristics of the clustered application and its shutdown procedure, it could be a long time before the shutdown is successful (minutes to hours). For instance, many applications have the concept of “draining,” where they accept no new work, but complete any work in progress before exiting. This process may take a long time if, for instance, a long-running transaction is active.

When VxVM’s shutdown procedure is invoked, it checks all volumes in all shared disk groups on the node that is being shut down and then either proceeds with the shutdown or fails:

- If all volumes in shared disk groups are closed, VxVM makes them unavailable to applications. Since all nodes know that these volumes are closed on the leaving node, no resynchronizations are performed.
- If any volume in a shared disk group is open, the VxVM shutdown procedure returns failure. The shutdown procedure can be retried repeatedly until it succeeds. There is no timeout checking in this operation — it is intended as a service that verifies that the clustered applications are no longer active.

Note Once shutdown has succeeded, the node has left the cluster. It is not possible to access the shared volumes until it joins the cluster again.

Since shutdown may be a lengthy process, other reconfigurations may take place while shutdown is in progress. Normally, the shutdown attempt is suspended until the other reconfiguration completes. However, if it is already too far advanced, the shutdown may complete first.

Node Abort

If a node does not leave cleanly, this is either because the host crashed or because some cluster component decided to make the node leave on an emergency basis. The ensuing cluster reconfiguration calls the VxVM abort function. This function makes an attempt to halt all access to shared volumes at once, though the operation does wait until I/O that is at the disk completes.

I/O operations that have not yet been started are failed, and the shared volumes are removed. Applications that were accessing the shared volumes therefore fail with errors.

After a node abort or crash, the shared volumes must be recovered (either by a surviving node or by a subsequent cluster restart) because it is very likely that there are unsynchronized mirrors.

Cluster Shutdown

When all the nodes in the cluster leave, the determination of whether or not the shared volumes should be recovered has to be made at the next cluster startup. If all nodes left cleanly, there is no need for recovery. If the last node left cleanly and resynchronization resulting from the non-clean leave of earlier nodes was complete, there is also no need for recovery. However, recovery must be performed if the last node did not leave cleanly or if resynchronization from previous leaves was not complete.

Disks in VxVM Clusters

The nodes in a cluster must always agree on the status of a disk. In particular, if one node cannot write to a given disk, all nodes must stop accessing that disk before the results of the write operation are returned to the caller. Therefore, if a node cannot contact a disk, it should contact another node to check on the disk's status. If the disk has failed, no node will be able to access it and the nodes can agree to detach the disk. If the disk has not failed, but rather the access paths from some of the nodes have failed, the nodes cannot agree on the status of the disk. A policy must exist to resolve this type of discrepancy.

Disk Detach Policies

In order to address the above discrepancy, the following policies (set for a diskgroup) are provided. These can be set by using the `vxedit` (1M) command.

Under the global connectivity policy for shared disk group(s), the detach occurs cluster-wide (globally), if any node in the cluster reports a disk(s) failure. This is the default policy.

Under local connectivity policy, in the event of disk(s) failing, the failures are confined to the particular node(s) which saw the failure. Note that an attempt is made to communicate with all nodes in the cluster to ascertain the disk(s) usability. If all nodes report a problem with the disk(s), a cluster-wide detach occurs.

Disk Group Activation

Disk group activation controls volume I/O capability from different nodes in the cluster. It is not possible to activate a diskgroup on a given node if it is activated in a conflicting mode on another node in the cluster.

Table6, ”Allowed and Conflicting Activation Modes,” summarizes the allowed and conflicting activation modes for shared disk groups, as follows:

Table 6. Allowed and Conflicting Activation Modes

Disk group activated in the cluster as....	**Attempt to activate disk group on another node as....**			
	Exclusive write	Shared write	Read only	Shared read
Exclusive write	Fail	Fail	Fail	Succeed
Shared write	Fail	Succeed	Fail	Succeed
Read only	Fail	Fail	Succeed	Succeed
Shared read	Succeed	Succeed	Succeed	Succeed

To place activation modes under user control, a defaults file `/etc/default/vxdg` must be created, and contain the following line:

```
default_activation_mode=activation-mode
```

where activation mode is: *off*, *shared-write*, *shared-read*, *read-only*, or *exclusive-write*.

When a shared disk group is created or imported, it will be activated in the specified mode. When a node joins the cluster, all shared disk groups will be activated in the specified mode.

Notes:

- When enabling activation using the defaults file, it is recommended that the defaults file be identical on all nodes in the cluster. Otherwise, the results of activation are unpredictable.
- If the default activation node is anything other than off, an activation following a cluster join, or a disk group creation or import may fail if another node in the cluster has activated the disk group in a conflicting mode.

Dirty Region Logging and Cluster Environments

Dirty Region Logging (DRL) is an optional property of a volume that provides speedy recovery of mirrored volumes after a system failure. Dirty Region Logging is supported in cluster-shareable disk groups. This section provides a brief overview of DRL and describes how DRL behaves in a cluster environment.

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. DRL logically divides a volume into a set of consecutive regions and maintains a dirty region log that contains a status bit representing each region of the volume. *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, which is associated with one of the volume's plexes.

Before writing any data to the volume, the regions being written are marked dirty in the log. 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. A log region becomes clean again after the write to the mirror has completed. On system restart, the Volume Manager recovers only those regions of the volume which are marked as dirty in the dirty region log.

In a cluster environment, the Volume Manager's implementation of DRL differs slightly from the normal implementation. The following sections outline some of the differences and discuss some aspects of the cluster environment implementation.

Log Format and Size

As with VxVM in the non-clustered case, the clustered dirty region log exists on a log subdisk in a mirrored volume.

A VxVM dirty region log has a recovery map and a single active map. A clustered dirty region log, however, has one recovery map and multiple active maps (one for each node in the cluster). Unlike VxVM, the cluster feature places the recovery map at the beginning of the log.

The clustered dirty region log size is typically larger than a VxVM dirty region log, as it must accommodate active maps for all nodes in the cluster plus a recovery map. The size of each map within the dirty region log is one or more whole blocks. `vxassist` automatically takes care of allocating a sufficiently large dirty region log.

The log size depends on the volume size and the number of nodes. The log must be large enough to accommodate all maps (one map per node plus a recovery map). Each map should be one block long for each two gigabytes of volume size. For a two-gigabyte volume in a two-node cluster, a log size of three blocks (one block per map) should be sufficient; this is the minimum log size. A four-gigabyte volume in a four-node cluster requires a log size of ten blocks, and so on.

When nodes are added to an existing cluster, the existing DRL logs need to be detached and removed (using `vxplex -o rm dis`) and then recreated (using `vxassist addlog`). This increases the log sizes so that they can accommodate maps for the additional nodes.

Compatibility

Except for the addition of a cluster-specific magic number, DRL headers in a cluster environment are the same as their non-clustered counterparts.

It is possible to import a VxVM disk group (and its volumes) as a shared disk group in a cluster environment and vice versa. However, the dirty region logs of the imported disk group may be considered invalid and a full recovery may result.

If a shared disk group is imported by a VxVM system without cluster support, VxVM will consider the logs of the shared volumes to be invalid and will conduct a full volume recovery. After this recovery completes, the Volume Manager will use the cluster feature's Dirty Region Logging.

The Volume Manager cluster feature is capable of performing a DRL recovery on a non-shared VxVM volume. However, if a VxVM volume is moved to a VxVM system with cluster support and imported as shared, the dirty region log will probably be too small to accommodate all the nodes in the cluster. The cluster feature will therefore mark the log invalid and do a full recovery anyway. Similarly, moving a DRL volume from a two-node cluster to a four-node cluster may result in too small a log size, which the cluster feature will handle with a full volume recovery. In both cases, the system administrator is responsible for allocating a new log of sufficient size.

How DRL Works in a Cluster Environment

When one or more nodes in a cluster crash, DRL needs to be able to handle the recovery of all volumes in use by those nodes when the crash(es) occurred. On initial cluster startup, all active maps are incorporated into the recovery map; this is done during the `volume start` operation.

Nodes that crash (i.e., leave the cluster as "dirty") are not allowed to rejoin the cluster until their DRL active maps have been incorporated into the recovery maps on all affected volumes. The recovery utilities compare a crashed node's active maps with the recovery map and make any necessary updates before the node can rejoin the cluster and resume I/O to the volume (which overwrites the active map). During this time, other nodes can continue to perform I/O.

The VxVM kernel tracks which nodes have crashed. If multiple node recoveries are underway in a cluster at a given time, their respective recoveries and recovery map updates can compete with each other. The VxVM kernel therefore tracks changes in the DRL recovery state and prevents I/O operation collisions.

The master performs volatile tracking of DRL recovery map updates for each volume and prevents multiple utilities from changing the recovery map simultaneously.

Upgrading Volume Manager Cluster Functionality

The rolling upgrade feature allows an administrator to upgrade the version of Volume Manager running in a cluster without shutting down the entire cluster. To install the new version of Volume Manager running on a cluster, the system administrator can pull out one node from the cluster, upgrade it and then join the node back into the cluster. This is done for each node in the cluster.

Every VxVM release, starting with Release 3.1, has a *cluster protocol version* number associated with it. This is different from the release number. The cluster protocol version is stored in the `/etc/vx/volboot` file. In a new installation of VxVM, the `volboot` file does not exist in the `/etc/vx` directory. `vxdctl init` creates this file and sets the cluster protocol version to the highest supported version.

A new VxVM release supports two versions of cluster protocol. The lower version corresponds to the existing VxVM release. This has a fixed set of features and communication protocols. The higher version corresponds to a new release of VxVM which has a new set of these features. If the new release of VxVM does not have any functional or protocol changes, the cluster protocol version remains unchanged, for example, in case of bug fixes or minor changes. In this case, `vxdctl upgrade` need not be executed.

During the rolling upgrade operation each node must be shut down and the VxVM release with the latest cluster protocol version must be installed. All nodes that have the new release of VxVM continue to use the lower level version. A slave node that has the new cluster protocol version installed tries to join the cluster. If the new cluster protocol version is not in use on the master node, it rejects the join and provides the current cluster protocol version to the slave node. The slave retries the join with the cluster protocol version provided by the master node. If the join fails at this point, the cluster protocol version on the master node is out of range of the protocol versions supported by the joining slave. In such a situation the system administrator must upgrade the cluster through each intermediate release of VxVM to reach the latest supported cluster protocol version.

All nodes are upgraded to the latest cluster protocol version and the new features are available.

Once all nodes have the new release installed, the `vxdctl upgrade` command must be run on the Master node to switch to the higher cluster protocol version.

Cluster-related Volume Manager Utilities and Daemons

The following utilities and/or daemons have been created or modified for use with the Volume Manager in a cluster environment:

- `vxclust`
- `vxclustadm`
- `vxconfigd`
- `vxdg`
- `vxdisk`
- `vxrecover`
- `vxdctl`
- `vxstat`

The following sections contain information about how each of these utilities is used in a cluster environment. For further details on any of these utilities, refer to their manual pages.

vxclust

Every time there is a cluster reconfiguration, every node currently in the cluster runs the `vxclust` utility at each of several well-orchestrated steps. Cluster manager facilities ensure that the same step is executed on all nodes

Notes:

- Most Volume Manager commands require superuser privilege.
- vxclust works with SunCluster™ as cluster manager.

[illegible] `vxclust` determines what the Volume Manager cluster feature should do next. After informing

If a node does not respond to a `vxclust` request within a specific timeout period, that node aborts. `vxclust` then decides whether to restart the reconfiguration or give up, depending on the circumstances. If the cause of the reconfiguration is a local, uncorrectable error, `vxclust` gives up. If a node cannot complete an operation because another node has left, the surviving node times out. In this case, `vxclust` requests a reconfiguration with the expectation that another node will leave. If no other node leaves, `vxclust` will cause the local node to leave.

If a reconfiguration step fails, `vxclust` returns an error to the cluster manager. The cluster manager may decide to abort the node, causing its immediate departure from the cluster. Any I/O in progress to the shared disk fails and access to the shared disks is stopped.

`vxclust` decides what actions to take when it is informed of changes in the cluster. If a new master node is required (due to failure of the previous master), `vxclust` determines which node becomes the new master.

vxclustadm

Note vxclustadm works only with VCS.

The vxclustadm command activates and deactivates cluster functionality of VxVM on a node in the cluster. It is called from online and offline scripts during VCS cluster startup and shutdown.

The startnode option passes the cluster configuration information to the VxVM kernel. In response to this command, the kernel and the configuration daemon, vxconfigd, perform initialization.

The stopnode option stops cluster functionality on a node. It waits for all outstanding I/O to complete and all applications to close shared volumes. The abortnode option aborts the clustering activity on a node.

This is an emergency shutdown that aborts all the uncompleted I/O on shared volumes.

The nodestate option determines the state of a node in the cluster.

Refer to vxclustadm(1M) for more information.

vxconfigd

The Volume Manager configuration daemon, vxconfigd, maintains configurations of VxVM objects. vxconfigd receives cluster-related instructions from the vxclust utility under SunCluster or the kernel when running VCS. A separate copy of vxconfigd resides on each node; these copies communicate with each other through networking facilities. For each node in a cluster, Volume Manager utilities communicate with the vxconfigd running on that particular node; utilities do not attempt to connect with vxconfigd daemons on other nodes. During startup of the cluster, vxclust (SunCluster) or the kernel (VCS) tells vxconfigd to begin cluster operation and tells it whether it is a master or slave node.

When a node is initialized for cluster operation, vxconfigd is notified that the node is about to join the cluster and is provided with the following information (from the cluster manager configuration database):

- the cluster ID
- the node IDs
- the master node ID
- the node’s role
- the network address of the vxconfigd on each node

On the master node, vxconfigd sets up the shared configuration (i.e., imports the shared disk groups) and informs vxclust (SunCluster) or the kernel (VCS) when it is ready for slaves to join.

On slave nodes, vxconfigd is notified when the slave node can join the cluster. When the slave node joins the cluster, vxconfigd and the Volume Manager kernel communicate with their counterparts on the master in order to set up the shared configuration.

When a node leaves the cluster, the vxconfigd daemon notifies the kernel on all the other nodes. The master node then performs any necessary cleanup. If the master node leaves the cluster, the kernels choose a new master node and the vxconfigd daemons on all nodes are notified of the choice.

vxconfigd also participates in volume reconfiguration. See “Volume Reconfiguration” on page 198 for information on vxconfigd’s role in volume reconfiguration.

vxconfigd Recovery

The Volume Manager vxconfigd daemon may be stopped and/or restarted at any time. While vxconfigd is stopped, volume reconfigurations cannot take place and other nodes cannot join the cluster until vxconfigd is restarted. In the cluster, the vxconfigd daemons on the slaves are always connected to the vxconfigd daemon on the master. It is therefore not advisable to stop the vxconfigd daemon on any clustered node.

If vxconfigd is stopped for some reason, different actions are taken depending on which node has a stopped daemon:

- If vxconfigd is stopped on the slave(s), the master takes no action. When vxconfigd is restarted on the slave, the slave’s vxconfigd attempts to reconnect to the master’s and re-acquire the information about the shared configuration. (The kernel’s view of the shared configuration is unaffected, and so is access to the shared disks.) Until the slave vxconfigd has successfully rejoined to the master, it has very little information about the shared configuration and any attempts to display or modify the shared configuration may fail. In particular, if the shared disk groups are listed (using vxdg list), they will be marked as disabled; when the rejoin has completed successfully, they will be marked as enabled.
- If vxconfigd is stopped on the master, vxconfigd on the slave(s) attempts to rejoin to the master periodically. This will not succeed until vxconfigd is restarted on the master. In this case, the slave vxconfigd’s information about the shared configuration has not been lost, so configuration displays are accurate.
- If vxconfigd is stopped on both the master and the slave(s), the slave will not display accurate configuration information until vxconfigd has been restarted on both nodes and they have reconnected again.

When vxclust (SunCluster) or the kernel (VCS) notices that vxconfigd is stopped on a node, vxconfigd is restarted.

Note With VxVM, the `-r reset` option to `vxconfigd` restarts `vxconfigd` and creates all states from scratch. This option is not available while a node is in the cluster because it would cause the loss of cluster information; if this option is used under these circumstances, `vxconfigd` will not start.

vxdg

The `vxdg` utility manages Volume Manager disk groups. `vxdg` can be used to specify that a disk group is cluster-shareable. The `-s` option to `vxdg` is provided to initialize or import a disk group as "shared."

If the cluster software has been run to set up the cluster, a shared disk group can be created with the following command:

```
vxdg -s init diskgroup [medianame=]accessname
```

where *diskgroup* is the disk group name; *medianame* is the administrative name chosen for the disk; and *accessname* is the disk access name (or device name).

Disk groups can be imported as shared using `vxdg -s import`. If the disk groups were set up before the cluster software was run, the disk groups can be imported into the cluster arrangement with the command:

```
vxdg -s import diskgroup
```

where *diskgroup* is the disk group name or ID. On subsequent cluster restarts, the disk group will automatically be imported as shared. Note that it may be necessary to deport the disk group (using `vxdg deport` *diskgroup*) before invoking this command.

A disk group can be converted from shared to private by deporting it via `vxdg deport` and then importing it with `vxdg import` *diskgroup*.

Note The system cannot tell if a disk is shared. To protect data integrity when dealing with disks that can be accessed by multiple systems, the system administrator must be careful to use the correct designation when adding a disk to a disk group. If the administrator attempts to add a disk that is not physically shared to a shared disk group, the Volume Manager allows this on the node where the disk is accessible if that node is the only one in the cluster. However, other nodes will not be able to join the cluster. Furthermore, if the administrator attempts to add the same disk to different disk groups on two nodes at the same time, the results are undefined. All configurations should therefore be handled on one node only.

`vxdg` has a force option (`-f`) that can be used to force-import a disk group or force-add a disk to a disk group.

Note The force option(`-f`) should be used with caution and should only be used if the system administrator is fully aware of the possible consequences.

When a cluster is restarted, VxVM may refuse to auto-import a disk group for one of the following reasons:

- A disk in that disk group is no longer accessible because of hardware errors on the disk. In this case, the system administrator can reimport the disk group with the force option as follows:

  ```
  # vxdg -s -f import diskgroup
  ```

- Some of the nodes to which disks in the disk group are attached are not currently in the cluster, so the disk group cannot access all of its disks. In this case, a forced import is unsafe and should not be attempted (because it can result in inconsistent mirrors).

If VxVM will not add a disk to an existing disk group (because that disk is not attached to the same node(s) as the other disks in the disk group), the system administrator can force-add the disk as follows:

```
# vxdg -f adddisk -g diskgroup [medianame=]accessname
```

`vxdg` can also be used to list shared disk groups. The following command displays one line of information for each disk group:

```
# vxdg list
```

The output from this command is as follows:

```
NAME         STATE             ID
rootdg       enabled           774215886.1025.teal
group2       enabled,shared    774575420.1170.teal
group1       enabled,shared    774222028.1090.teal
```

Shared disk groups are designated with the flag `shared`.

The following command displays one line of information for each shared disk group:

```
# vxdg -s list
```

The output for this command is as follows:

```
NAME         STATE            ID
group2       enabled,shared   774575420.1170.teal
group1       enabled,shared   774222028.1090.teal
```

The following command shows information about one specific disk group, including whether it is shared or not:

```
# vxdg list diskgroup
```

where *diskgroup* is the disk group name.

The output for `vxdg list group1` on the master (for the disk group `group1`) is as follows:

```
Group:     group1
dgid:      774222028.1090.teal
import-id: 32768.1749
flags:     shared
copies:    nconfig=default nlog=default
config:   seqno=0.1976 permlen=1456 free=1448 templen=6 loglen=220
config disk c1t0d0s2 copy 1 len=1456 state=clean online
config disk c1t1d0s2 copy 1 len=1456 state=clean online
log disk c1t0d0s2 copy 1 len=220
log disk c1t1d0s2 copy 1 len=220
```

Note that the `flags:` field is set to `shared`. The output for the same command is slightly different on a slave.

vxdisk

The `vxdisk` utility manages Volume Manager disks. `vxdisk` can be used to determine whether a disk is part of a cluster-shareable disk group, as follows:

```
# vxdisk list accessname
```

where *accessname* is the disk access name (or device name).

The output from this command (for the device `c1t0d0s2`) is as follows:

```
Device:    c1t0d0s2
devicetag: c1t0d0
type:      sliced
clusterid:    cvm
disk:      name=disk01 id=774215890.1035.teal
group:     name=group1 id=774222028.1090.teal
flags:     online ready autoconfig shared imported
pubpaths:  block=/dev/dsk/c1t0d0s4 char=/dev/rdsk/c1t0d0s4
privpaths: block=/dev/dsk/c1t0d0s3 char=/dev/rdsk/c1t0d0s3
version:   3.1
iosize:    min=512 (bytes) max=248 (blocks)
public:    slice=4 offset=0 len=2050272
private:   slice=3 offset=1 len=2015
update:    time=778564769 seqno=0.1614
headers:   0 248
configs:   count=1 len=1456
logs:      count=1 len=220
Defined regions:
 config   priv 000017-000247[000231]: copy=01 offset=000000 enabled
 config   priv 000249-001473[001225]: copy=01 offset=000231 enabled
 log     priv 001474-001693[000220]: copy=01 offset=000000 enabled
```

Note that the `clusterid:` field is set to `cvm` (the name of the cluster) and the `flags:` field includes an entry for `shared`. When a node is not joined, the `flags:` field contains the `autoimport` flag instead of `imported.`

vxrecover

The `vxrecover` utility recovers plexes and volumes after disk replacement.

When a node leaves the cluster, it may leave some mirrors in an inconsistent state. The `vxrecover` utility performs recovery on all volumes in this state. The `-c` option causes `vxrecover` to perform recovery for all volumes in cluster-shareable disk groups. `vxclust` automatically calls `vxrecover -c`, when necessary.

Note While `vxrecover` is active, there may be some degradation in system performance.

vxdctl

The `vxdctl` utility manages some aspects of the volume configuration daemon, `vxconfigd.`The `-c` option can be used to request cluster information. `vxdctl` can be used as follows to determine whether `vxconfigd` is enabled and/or running:

```
# vxdctl -c mode
```

Depending on the circumstances, this displays output similar to the following:

```
mode: enabled: cluster active - MASTER
mode: enabled: cluster active - SLAVE
mode: enabled: cluster inactive
mode: enabled: cluster active - role not set
```

Note If `vxconfigd` is disabled, no cluster information is displayed.

Refer to the `vxdctl`(1M) man page for a complete description of `vxdctl`.

`vxdctl` lists the cluster protocol version and cluster protocol range. After all the nodes in the cluster are updated with the new cluster protocol, the entire cluster is upgraded with the following command:

```
# vxdctl upgrade
```

The `vxdctl protocolversion` command is used to check the existing cluster protocol version. For example:

```
# vxdctl protocolversion
Cluster running at protocol 10
```

The `vxdctl protocolrange` command displays the maximum and minimum cluster protocol version supported by the current VxVM release, for example,

```
# vxdctl protocolrange
minprotoversion: 10, maxprotoversion: 20
```

The `vxdctl list` command displays the cluster protocol version running on a node. The following is an example of the output of the `vxdctl list` command:

```
Volboot file
version: 3/1
seqno: 0.19
cluster protocol version: 20
hostid: giga
entries:
```

The `vxdctl support` command displays the maximum and minimum protocol version supported by the node and the current protocol version. The following is an example of the output of the `vxdctl support` command:

```
Support information:
  vold_vrsn: 11
  dg_minimum: 60
  dg_maximum: 70
  kernel: 10
  protocol_minimum: 10
  protocol_maximum: 20
  protocol_current: 20
```

vxstat

`vxstat` returns statistics for specified objects. In a cluster environment, `vxstat` gathers statistics from all of the nodes in the cluster. The statistics give the total usage, by all nodes, for the requested objects. If a local object is specified, its local usage is returned.

`vxstat` allows the caller to optionally specify a subset of nodes:

```
# vxstat -g diskgroup -n node[,node...]
```

where *node* is an integer. If a comma-separated list of nodes is supplied, `vxstat` displays the sum of the statistics for the nodes in the list.

In the following example, `vxstat` is instructed to obtain statistics for node 2, volume `vol1`:

```
# vxstat -g group1 -n 2 vol1
```

This might produce output similar to the following:

```
                    OPERATIONS          BLOCKS          AVG TIME(ms)
TYP  NAME          READ     WRITE      READ     WRITE    READ  WRITE
vol  vol1          2421         0    600000         0    99.0    0.0
```

`vxstat` can also obtain and display statistics for the entire cluster, as follows:

```
# vxstat -b
```

The statistics for all nodes are added together. For example, if node 1 did 100 I/Os and node 2 did 200 I/Os, `vxstat -b` would return `300`.

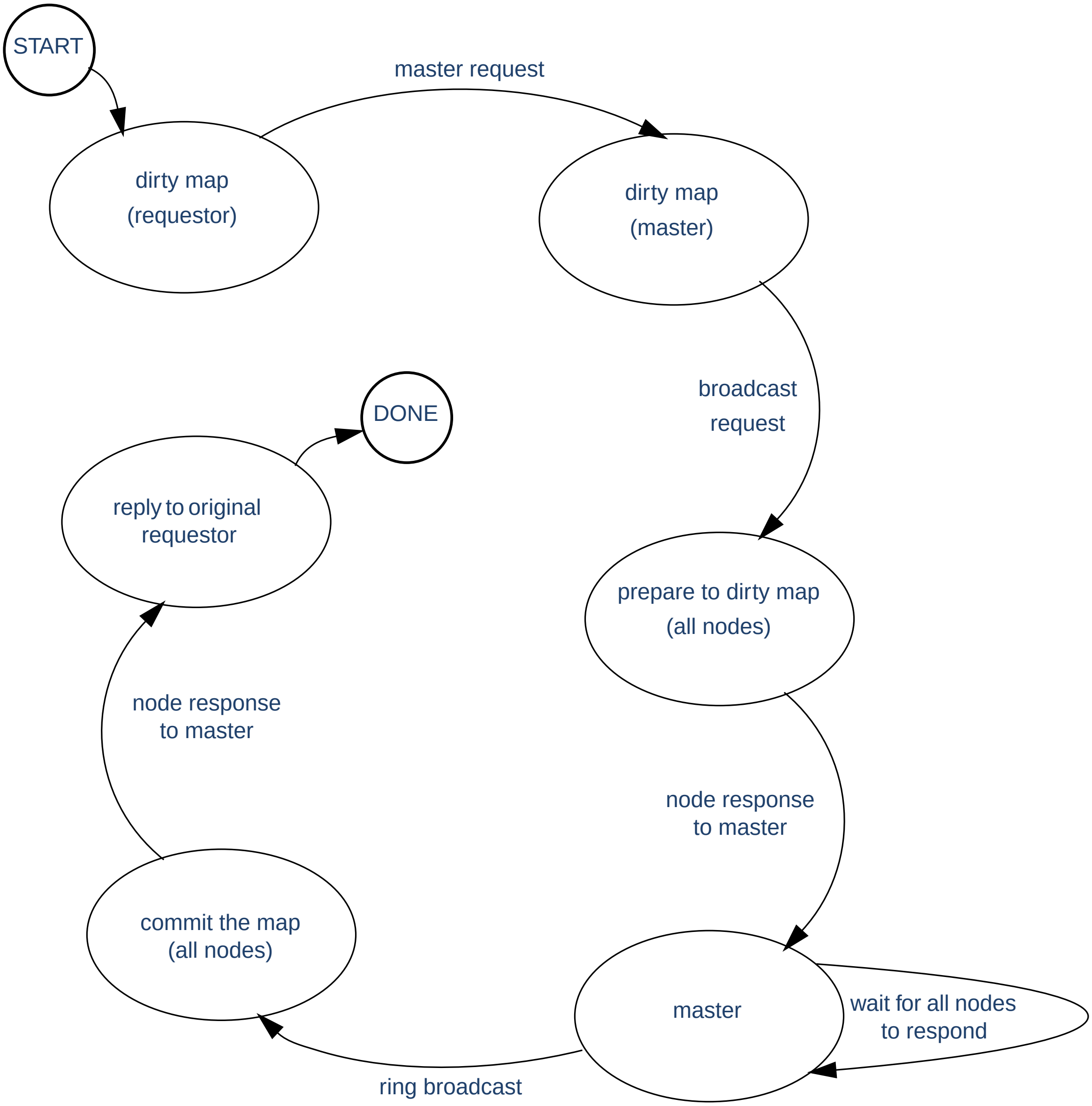
Clustering and FastResync

FastResync) is supported for shared volumes. The update FastResync maps are distributed across in the cluster. The existence of non-persistent maps is less moot in a cluster environment because only one node in the cluster must be up for the FastResync maps to be available. A single node crash does not cause the loss of FastResync maps.

Since the region size must be the same on all nodes in a cluster for a shared volume, the value of the *vol_fmr_logsz* tunable on the master overrides the tunable value on slaves (if they are different). Also, since the master may change for a shred volume, this value is retained for the life of the volume, or until FMR is turned on for the volume

The map updates are applied under the auspices of the master node. When the master node distributes updates to all nodes, all updates are applied either to all nodes or to none of the nodes. The master node orchestrates a two-phase commitment for applying any updates. See Figure 25, ”Bitmap Clusterization,” on page 215.

Figure 25. Bitmap Clusterization



Recovery 7

Introduction

The VERITAS Volume Manager protects systems from disk failures and helps you to recover from disk failures. This chapter describes 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 about protecting your system, see “Volume Manager Initialization” on page 47 and “System Setup” on page 50.

The following topics are covered in this chapter:

- The UNIX Boot Process
- Possible Root (/), swap, and usr Configurations
- Failures and Recovery Procedures
- Hot-Relocation and Boot Disk Failures
- Re-Adding and Replacing Boot Disks
- Reattaching Disks
- Reinstallation Recovery
- Plex and Volume States
- RAID-5 Volume Recovery
- Miscellaneous RAID-5 Operations

The UNIX Boot Process

A Sun SPARC system prompts for the `boot` command unless the `autoboot` flag has been set in the nonvolatile storage area used by the firmware. Machines with older PROMs have different prompts than the prompt for the newer V2 and V3 versions of PROM. 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 Volume Manager, you can use boot disk alias names. These aliases can take the form of Volume Manager provided names (for example, `vx-rootdisk` or `vx-disk01`) or operating system provided names (for example, `disk1`). You can view a list of possible bootable devices by using this command at the OpenBoot `OK` prompt: `devalias.`

The *filename* is the name of a standalone program to the `boot` program. The default is to boot `/kernel/unix` from the `root` partition. You can specify another program (such as `/stand/diag`) on the command line. Some versions of the firmware allow the default filename to be saved in the nonvolatile 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 Volume Manager with rootability does not boot with the defaults presented by the `-a` flag. See "/etc/system Copy Available" on page 226 for the correct responses to `boot -a`.

Flags are not interpreted by the `boot` program. The `boot` program passes all boot-flags to the file identified by *filename*. Boot. See the `kernel` (1) and `kadb` (1M) manual pages for information on the options available with the default standalone program, `/kernel/unix`.

Booting After Failures

If the root disk is mirrored, you can use the alternate boot disk to boot the system if the primary boot disk fails. To boot the system after failure of the primary boot disk, follow these steps:

1. Check for aliased VM disks using the `devalias` command at the OpenBoot command prompt.

   Disks that are suitable mirrors of the root disk are listed with the name : `vx` *medianame*, where *medianame* is the disk media name for the disk with the candidate root file system.

2. Enter this command:

```
# ok boot alias_name
```

where *alias_name* is 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 lists any nonstale alternate bootable disks.

Possible Root (/), swap, and usr Configurations

During installation, different configurations are possible for root, swap, and usr file systems. These 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 Volume Manager control.
- usr is on a separate partition that is on the root disk. In this case, a separate volume is created for the usr partition. vxmirror mirrors 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 is created for the usr partition only if that disk is encapsulated by Volume Manager. Note that in this case, encapsulating the root disk and having mirrors of the root volume does not help if the usr partition becomes inaccessible for any reason. It is recommended that you 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. See “Boot-time Volume Restrictions” on page 75 for information on rootvol and usr volume restrictions.

Volume Manager allows 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 Volume Manager 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 eventually causes the system to crash. It may be possible to boot the system, but having mirrors for the swapvol volume prevents system failures.

Repairing Root (/) or /usr File Systems on Volumes

If the root (`/`) or `/usr` file system becomes unusable, it is desirable to boot from a network-mounted root file system or from a valid backup. The backup must have all relevant file system partitions on the root disk. Also, you need a printout of the root disk partition table before the root disk was encapsulated.

This task is made more difficult when the `root` or `/usr` file system is defined on a mirrored volume. 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:

- 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 is to restore the system from a valid backup tape. The procedure is described below. This procedure does not require the operating system to be installed from the base CD-ROM.

The procedure described below provides a simple, efficient, and reliable means of recovery when both the root disk and its mirror are damaged.

Recovering a Volume Manager Root Disk (and Root Mirror) From Backup Tape

This procedure assumes that you have:

- A current full backup of all the file systems on the original Volume Manager root disk.
- A new boot disk installed to replace the original failed boot disk if the original boot disk was physically damaged.

This procedure requires the reinstallation of the Volume Manager root disk. To prevent the loss of data on disks not involved in the reinstallation, you should only involve the Volume Manager root disk in the reinstallation procedure.

Several of the automatic options for installation access disks other than the root disk without requiring confirmation from the administrator. Therefore, it is advisable that you disconnect all other disks (containing volumes) from the system prior to starting this procedure. Disconnecting the other disks ensures that they are unaffected by the reinstallation. Reconnect these disks after the procedure has been completed.

Recovering a Root Disk

In the proceedure below, assume that the (new) boot disk is `c0t0d0` and that you need to recover both the `/` and `/usr` file systems, `s0` and `s6` respectively.

1. Boot the operating system from the CDROM.

2. Use the `format` command to create identical partitions on the (new) boot disk (`c0t0d0`) to contain the file systems previously on the original boot disk.

Note Since you need to restore the file systems, you can have a maximum of seven partitions. Additionally, as you need to re-encapsulate this disk momentarily, you can only have a maximum of five partitions as two are required for the private and public regions on the disk.

3. Mount `/dev/rdsk/c0t0d0s0` on `/a/root.` Restore the root file system from tape. Install a bootblock device on `/a/root` using `installboot.`

4. Mount `/dev/rdsk/c0t0d0s6` on `/a/usr.` Restore the `/usr` file system from tape.

5. Modify the restored root file system as follows:

   - `touch /a/root/etc/vx/reconfig.d/state.d/install-db`
   - modify `/a/root/etc/system` by removing these two lines:

     ```
     rootdev:/pseudo/vxio@0:0
     set vxio:vol_rootdev_is_volume=1
     ```

   - modify `/a/root/etc/vfstab` by replacing the Volume Manager volume device entries `/dev/vx/dsk` with the standard disk devices `/dev/dsk/c0t0d0s0` and `/dev/dsk/c0t0d0s6`.

6. Reboot the system from the (new) boot disk. This allows the boot disk to come up thinking that Volume Manager is *not* installed.

The next step in the procedure depends on whether there are root disk mirrors in the old `rootdg`:

- If there are other disks in the old `rootdg` that are *not* used as root disk mirrors, if so, go to step 7.
- If there are only root disk mirrors in the old `rootdg`, if so, go to step 8.

7. If there are other disks in the old `rootdg` that are *not* used as root disk mirrors, follow these steps to bring in the old `rootdg` (minus the boot disk which Volume Manager will think has failed) and set up the new boot disk.

   a. Remove files involved with the installation that are no longer needed:

      ```
      rm -r /etc/vx/reconfig.d/state.d/installdb
      ```

b. Start the VxVM I/O daemons:

```
# vxiod set 10
```

c. Start the VxVM configuration daemon in disabled mode:

```
# vxconfigd -m disable
```

d. Initialize the `vxconfigd` daemon:

```
# vxdctl init
```

e. Enable `vxconfigd`:

```
# vxdctl enable
```

The above steps should bring in the old rootdg minus the root disk which VxVM will think has failed.

a. Use the `vxedit` command (or the Volume Manager Storage Administrator) to remove the old root disk volumes and the root disk itself.

b. Use the `vxdiskadm` command to encapsulate the (new) boot disk and initialize any disks that are to serve as root disk mirrors. After the required reboot, mirror the root disk to the root disk mirrors.

c. If there are only root disk mirrors in the old `rootdg`:

   Run the `vxinstall` command to encapsulate the (new) boot disk, and initialize the root disk mirror(s).

   After the required reboot, mirror the root disk to the root disks mirrors.

Backing Up and Restoring the Root File System

It is a good idea to 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 using this command:

```
# /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 “Repairing Root (/) or /usr File Systems on Volumes” on page 220).

2. Mount and restore the root file system by using these commands:

```
   # newfs /dev/vx/rdsk/rootvol
   # mount /dev/vx/dsk/rootvol /mnt
#  cd /mnt
   # /usr/lib/fs/ufs/ufsrestore [restore-options]
```

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.

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 this message appears during the boot attempt, the system should be booted 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 are similar to this:

```
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 “Re-Adding a Failed Boot Disk” on page 229. However, if the reattach fails, then the disk needs to be replaced as described in “Replacing a Failed Boot Disk” on page 231.

Failures Accessing the Boot Device

Early in the boot process, immediately following system initialization, there may be messages similar to this:

```
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. In order 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 allow the system to boot, this also indicates hardware problems.

Failures Due to Incorrect Entries in /etc/vfstab

When the root disk is encapsulated and put under Volume Manager control, as part of the normal encapsulation process, volumes are created for all of the partitions on the disk. Volume Manager modifies the `/etc/vfstab` to use the corresponding volumes instead of the disk partitions. Care should be taken while editing the `/etc/vfstab` file manually. The most important entries are those corresponding to `/` and `/usr`. The `vfstab` that existed prior to Volume Manager installation is saved in `/etc/vfstab.prevm`.

Damaged / Entry in /etc/vfstab

If the entry in `/etc/vfstab` for `/` is lost or is incorrect, the system boots in single-user mode. Messages similar to the following are displayed:

```
File just loaded does not appear to be executable
```

It is recommended that you run fsck at this point:

```
# fsck /dev/vx/rdsk/rootvol
```

At this point in the boot process, / is not yet mounted read/write. Since the entry in /etc/vfstab was either incorrect or deleted, mount / as read/write manually, by using this command:

```
# mount -o remount /dev/vx/dsk/rootvol
```

After mounting / as read/write, exit the shell. The system prompts 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 /etc/vfstab for / after the system boots.

Damaged /usr Entry in /etc/vfstab

/etc/vfstab has an entry for /usr only if /usr is located on a separate disk partition. After encapsulation of the disk containing the /usr partition, Volume Manager changes the entry in /etc/vfstab to use the corresponding volume.

In the event of loss of the entry for /usr from /etc/vfstab, the system cannot be booted (even if you have mirrors of the /usr volume). In this case, boot the system from the CD-ROM and restore the /etc/vfstab. (See “Repairing Root (/) or /usr File Systems on Volumes” on page 220.)

Failures Due to Missing or Damaged /etc/system

Note Do not edit any entries in /etc/system that are added by Volume Manager. All Volume Manager entries are enclosed with *vxvm_START and *vxvm_END.

It is advisable to make a copy of /etc/system 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 prompts for the name of the system file, enter the path of the saved system file.

/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 Volume Manager rootability feature on. The system can be booted without Volume Manager rootability (i.e., without the rootvol being the /) if /usr is not a volume.

To boot the system without Volume Manager rootability, follow the steps outlined in “Repairing Root (/) or /usr File Systems on Volumes” on page 220 to:

1. Start the Volume Manager from the CD-ROM. See the *VERITAS Volume Manager Installation Guide* for more information.

2. Run the `fixmountroot` command.

3. Make and mount `/tmp/rootvol`.

You can then edit the file `/tmp/rootvol/etc/system` and do any other necessary repairs.

After booting the system, make the following entries in `/etc/system`:

```
* 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 file `/etc/system` so that it contains a line of the following form for each driver:

```
# forceload: drv/driver_name
```

The driver names for these disks can be obtained by doing a long listing on `/dev/dsk/`*root_device*. An example of a driver name is `io-unit`.

/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 Volume Manager rootability.

To boot the system with Volume Manager 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
```

/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, the system must be booted from the CD-ROM (using the steps outlined in "Repairing Root (/) or /usr File Systems on Volumes" on page 220). 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.

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 become stale. This means that the data on that disk is inconsistent 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 nonstale plexes. This problem can occur, for example, if the system was booted from one of the disks made bootable by Volume Manager with the original boot disk turned off. The system boots normally, but the plexes that reside on the unpowered disk are stale. If the system reboots from the original boot disk with the disk turned back on, the system boots using that stale plex.

Another possible problem can occur if errors in the Volume Manager headers on the boot disk prevent Volume Manager from properly identifying the disk. In this case, Volume Manager does 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 occur if the root disk has a failure that affects the root volume plex. At the next boot attempt, the system still expects to use the failed root plex for booting. If the root disk was mirrored at the time of the failure, an alternate root disk (with a valid root plex) can be specified for booting.

If any of these situations occur, the Volume Manager utility vxconfigd notes it when it is configuring the system as part of the init processing of the boot sequence. vxconfigd displays a message describing the error and what can be done about it, then it halts the system. For example, if the plex rootvol-01 of the root volume rootvol on disk rootdisk is stale, vxconfigd can display this 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 are 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 need to re-add or replace the disk.

If the plexes on the boot disk are unavailable, you should receive mail from Volume Manager 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 Volume Manager private region on the disk), vxdisk list shows this display:

```
DEVICE      TYPE        DISK        GROUP     STATUS
  -           -         rootdisk    rootdg    failed was: c0t3d0s2
c0t1d0s2     sliced     disk01      rootdg    ONLINE
```

Hot-Relocation and Boot Disk Failures

If the boot (root) disk fails and it is mirrored, 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 can fail for a root disk if the `rootdg` disk group does not contain sufficient spare or free space to fit 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 fails if these disks are not available.

Re-Adding and Replacing Boot Disks

Data that is not critical for booting the system is only accessed by the Volume Manager after the system is fully operational, so it does not have to be located in specific areas. The Volume Manager can find it. However, boot-critical data must be placed in specific areas on the bootable disks for the boot process to find it.

On some systems, the controller-specific actions done by the disk controller in the process and the system BIOS constrain the location of this critical data.

When a disk fails, there are two paths 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, reformatting a failed disk or doing a surface analysis to rebuild the alternate-sector mappings are sufficient to make a disk re-usable and a candidate for re-addition.
- If the disk has truly failed, it should be replaced.

The following sections describe how to re-add or replace a failed boot disk.

Re-Adding a Failed Boot Disk

Re-adding a disk is 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*. This means that Volume Manager has detected the disk failure and has ceased to access the disk.

Note Your system may use a *device name* or *path* that differs from the examples. See "Understanding Volume Manager" on page 17 for more information on device names.

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. This can be confirmed by listing the disks with the `vxdisk` utility with this command:

```
# vxdisk list
```

`vxdisk` displays this (example) list:

```
DEVICE        TYPE      DISK      GROUP      STATUS
c0t0d0s2      sliced    -         -          error
c0t1d0s2      sliced    disk02    rootdg     online
 -              -         disk01    rootdg     failed was:c0b0t0d0s0
```

Note that the disk `disk01` has no device associated with it, and has a status of `failed` with an indication of the device that it was detached from. It is also possible that the device *c0b0t0d0s0* would not be listed at all. This occurs if the disk fails totally and the disk controller does not detect it on the bus (for systems that use a bus).

In some cases, the `vxdisk list` output can differ. For example, if the boot disk has uncorrectable failures associated with the UNIX partition table. There can be a missing root partition that cannot be corrected but there are no errors in the Volume Manager private area. The output of the `vxdisk list` command displays this (example) list:

```
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 viewed as failed. In this case, it is necessary to 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 for more information about `vxdiskadm`). Once the disk is detached from the device, any special procedures for correcting the problem can be followed (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. Using the previous examples, you replace `disk01` with the device *c0t0d0s2* or *c0b0t0d0s2* (for systems that use a bus).

If hot-relocation is enabled when a mirrored boot disk fails, an attempt is made 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. At this point, `vxunreloc` can be invoked to move the hot-relocated subdisks back to the newly replaced disk.

Unrelocating Encapsulated Subdisks to a New Disk

When a boot disk is encapsulated, the root file system and other system areas, such as the swap devices on the boot disk, are made into volumes. Because Volume Manager uses part of the swap area to create the private region, it will be located in the middle of the disk. When a disk is initialized, instead of encapsulated, to become a Volume Manager disk, the private region is created at the beginning of the disk.

When an encapsulated boot disk fails, all of the subdisks are hot-relocated to other disks. As part of the relocation process, the original disk name and the offsets are stored in the subdisk records. After the failed boot disk is replaced with one that has the same storage capacity, it is "initialized" and added back to the disk group. `vxunreloc` can be run to automatically move all the subdisks back to the disk. However, the difference of the disk layout between an initialized disk and an encapsulated disk affects the way the offset into a disk is calculated for each unrelocated subdisk. You must use the `-f` option to move the subdisks to the disk, but not to the exact offset. If the replaced disk is larger than the original boot disk by at least 2M, then moving all the subdisks to the exact offsets on the disk can be successful.

`vxunreloc` makes the new disk bootable after it moves all the subdisks to the disk.

Note: A dump device is usually configured to be the swap partition of the root disk. Whenever a swap subdisk is moved (by hot-relocation, unrelocate, or manually) from one disk to another, the dump device must be re-configured to the new disk.

For Solaris 2.6: The dump device is stored in the `dumpfile` structure. To check the setting for the device, use the following command:

```
# echo dumpfile+0x10/s | adb -k /dev/ksyms /dev/mem
```

This setting can be changed by a reboot which configures the dump device to the first swap partiton.

For Solaris 2.7, and later:

The dump device can be viewed and set with the `dumpadm` command. For details, type:

```
# man dumpadm
```

Replacing a Failed Boot Disk

When a boot disk is to be replaced, the system should first be booted off an alternate boot disk. If the failing disk is not detached from its device, it should be manually detached using the "Remove a disk for replacement" function of `vxdiskadm.` See the `vxdiskadm` (1M) manual page for more information about `vxdiskadm`. Once the disk is detached, shut down the system and replace the hardware.

The replacement disk must have at least as much storage capacity as was in use on the disk being replaced. The replacement disk must 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, use the command:

```
# 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 system cleanly and replace the necessary hardware. When the hardware replacement is complete, boot the system. Use the vxdiskadm "Replace a failed or removed disk" function to replace the failing disk with the new device that was added.

Reattaching Disks

You can reattach a disk if it has 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 can use the vxreattach command to reattach the disks without plexes being flagged as stale. However, the reattach must occur 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 reattaches the failed disk media record to the disk with the same device name. The reattach occurs in the same disk group it was located in before and retains its original disk media name.

After a reattach takes place, recovery may not be necessary. The reattach can fail if the original (or another) cause for the disk failure still exists.

The command vxreattach -c checks whether a reattach is possible, but does not do 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.

Reinstallation Recovery

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, because there is currently no method of restoring the root file system from backup.

If these types of failures occur, 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.

General Reinstallation Information

This section describes procedures used to reinstall Volume Manager and preserve as much of the original configuration as possible after a failure.

System reinstallation destroys the contents of any disks that are used for reinstallation.

All Volume Manager related information is removed during reinstallation. Data removed includes data in private areas on removed disks that contain the disk identifier and copies of the Volume Manager configuration. The removal of this information makes the disk unusable as a Volume Manager disk.

The system root disk is always involved in reinstallation. Other disks can 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. Any other disks that are involved in the reinstallation, or that are removed and replaced, can lose Volume Manager configuration data (including volumes and mirrors).

If a disk, including the root disk, is not under Volume Manager control prior to the failure, no Volume Manager configuration data is lost at reinstallation. Any other disks can be replaced by following the procedures described under, “Disks and Disk Groups.”

Although it simplifies the recovery process after reinstallation, not having the root disk under Volume Manager control increases the possibility of a reinstallation being necessary. By having the root disk under Volume Manager 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. On some systems, the exceptions are the `root`, `stand`, and `usr` file systems, which cannot be restored from backup.

Reinstallation and Reconfiguration Procedures

To reinstall the system and recover the Volume Manager configuration, follow this 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 nonrelated Volume Manager packages.

3. Install Volume Manager.

   Add the Volume Manager package, but *do not* execute the `vxinstall` command.

4. Recover the Volume Manager configuration.

5. Clean up the Volume Manager configuration.

   This includes restoring any information in volumes affected by the failure or reinstallation, and recreating system volumes (`rootvol`, `swapvol`, `usr`, and other system volumes.).

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. It is advised that you 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 can still be recoverable. Removing the second disk ensures that the home file system remains intact.

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 Volume Manager.

You must be sure that no disks other than the root disk are accessed in any way while the operating system installation is in progress, If anything is written on a disk other than the root disk, the Volume Manager configuration on that disk can be destroyed.

Note During reinstallation, you can change the host ID (or name). It is recommended that you keep the existing host ID (name), as the sections that follow assume that you have not changed your host ID (name).

Reinstalling the Volume Manager

The installation of the Volume Manager has two parts:

- loading Volume Manager from CD-ROM
- initializing the Volume Manager

To reinstall the Volume Manager, follow the instructions for loading the Volume Manager (from CD-ROM) in the *VERITAS Volume Manager Installation Guide*.

Note To reconstruct the Volume Manager configuration left on the nonroot disks, *do not* initialize the Volume Manager (using `vxinstall`) after the reinstallation.

On some systems, you can use `vxserial` to install the Volume Manager license key (see the `vxserial` (1M) manual page).

Recovering the Volume Manager Configuration

Once the Volume Manager package has been loaded, recover the Volume Manager configuration by following these steps:

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 using this command:

   ```
   # shutdown -g0 -iS -y
   ```

5. When prompted, enter the password and press Return to continue.

6. Remove files involved with installation that were created when you loaded Volume Manager but are no longer needed. Use this command:

   ```
   # rm -rf /etc/vx/reconfig.d/state.d/install-db
   ```

7. After the files are removed, start some Volume Manager I/O daemons. Start the daemons by entering the command:

   ```
   # vxiod set 10
   ```

8. Start the Volume Manager configuration daemon, `vxconfigd`, in disabled mode by using this command:

   ```
   # vxconfigd -m disable
   ```

9. Initialize the `vxconfigd` daemon with this command:

   ```
   # vxdctl init
   ```

10. Initialize the DMP subsystem with this command:

    ```
    # vxdctl initdmp
    ```

11. Enable `vxconfigd` with this command:

    ```
    # 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 does not appear to the Volume Manager as a Volume Manager disk. The configuration of the preserved disks does not include the root disk as part of the Volume Manager 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. 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 `root` file system, `swap` area, (and on some systems `stand` area), and `/usr` file system are *no longer* located on volumes. To correct these problems, follow the instructions in "Configuration Cleanup."

The hot-relocation facility can be started after `vxdctl enable` succeeds, but should actually be started only when the Administrator is sure that its services, when enabled and operating, do not interfere with other reconfiguration procedures. It is recommended that hot-relocation be started after completion of the "Final Reconfiguration" steps. See "Starting Hot-Relocation" on page 242 for more information on starting hot-relocation.

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`, that contains the `root` file system
- `swapvol`, that contains the `swap` area
- (on some systems) `standvol`, that contains the `stand` file system
- `usr`, that contains the `/usr` file system

To remove the root volume, use the `vxedit` command:

```
# vxedit -fr rm rootvol
```

Repeat this command, using `swapvol` and `usr` (`standvol`) in place of `rootvol`, to remove the `swap`, `stand`, 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, do these steps:

1. Establish which VM disks have been removed or reinstalled by using this command:

   ```
   # vxdisk list
   ```

   The Volume Manager displays a list of system disk devices and the status of these devices. For example, for a reinstalled system with three disks and a reinstalled root disk, the output of the vxdisk list command is similar to this:

   ```
   DEVICE          TYPE      DISK      GROUP     STATUS
   c0t0d0s2       sliced    -         -         error
   c0t1d0s2       sliced    disk02    rootdg    online
   c0t2d0s2       sliced    disk03    rootdg    online
   -              -         disk01     rootdg    failed was:
   c0t0d0s2
   ```

Note Your system may use a device name that differs from the examples. See "Understanding Volume Manager" on page 17 for more information on device names.

   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. Use this command:

   ```
   # 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 returns 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. You use this command to print volume information:

   ```
   # 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 outputs this 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 v0101     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. The plex is no longer valid and must be removed.

Because `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 between 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 `vxprint -th v02` command displays:

```
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 earlier.

   A volume that has one mirror on a failed disk can 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) is similar to this example:

```
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  v0302     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 to create another plex later.

6. To remove an invalid plex, the plex must be dissociated from the volume and then removed. This is done with the `vxplex` command. To remove the plex `v03-02`, use this command:

   ```
   # vxplex -o rm dis v03-02
   ```

7. Once all the volumes have been cleaned up, clean up the disk configuration as described in “Disk Cleanup” on page 241.

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`, use this command:

```
# vxdg rmdisk disk01
```

If the `vxdg` command returns an error message, some invalid mirrors exist. Repeat the processes described in the section “Volume Cleanup” on page 237 until all invalid volumes and mirrors are removed.

Rootability Reconfiguration

Once all the invalid disks have been removed, the replacement or reinstalled disks can be added to Volume Manager control. If the root disk was originally under Volume Manager control or you now wish to put the root disk under Volume Manager control, add this disk first.

To add the root disk to Volume Manager control, use the Volume Manager Support Operations (`vxdiskadm`). Use this command:

```
# vxdiskadm
```

From the `vxdiskadm` main menu, select menu item `2 (Encapsulate a disk)`. Follow the instructions and encapsulate the root disk for the system.

When the encapsulation is complete, reboot the system to multi-user mode.

Final Reconfiguration

Once the root disk is encapsulated, any other disks that were replaced should be added using vxdiskadm. If the disks were reinstalled during the operating system reinstallation, they should be encapsulated; otherwise, they can be added.

Once all the disks have been added to the system, any volumes that were completely removed as part of the configuration cleanup can be recreated and their contents restored from backup. The volume recreation can be done by using vxassist or the graphical user interface.

You can recreate the volumes v01 and v02by using this vxassist command:

```
# vxassist make v01 24000
# vxassist make v02 30720 layout=stripe nstripe=3
```

Once the volumes are created, they can be restored from backup using normal backup/restore procedures.

Any volumes that had plexes removed as part of the volume cleanup can have these mirrors recreated by following the instructions for mirroring a volume with vxassist as described elsewhere in this manual. To replace the plex removed from volume v03 using vxassist, use this command:

```
# 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.

Starting Hot-Relocation

At this point, the Administrator should reboot the system or manually start up hot-relocation (if its service is desired). Either step causes the relocation daemon (and also its vxnotify process) to start.

To start hot-relocation, use the following commands.

Start the watch daemon. This sends E-mail to the administrator when any problems are found. To change the address used for sending problem reports, change the argument to vxrelocd:

```
# nohup /usr/lib/vxvm/bin/vxrelocd root &
```

The following command is also valid:

```
# nohup /usr/lib/vxvm/bin/vxrelocd root > /dev/null 2>&1 &
```

The following command can detect whether or not hot-relocation has been started:

```
# ps -ef | grep vxrelocd | grep -v grep
```

Plex and Volume States

The following sections describe plex and volume states.

Plex States

Plex states reflect whether or not plexes are complete and consistent copies (mirrors) of the volume contents. Volume Manager utilities automatically maintain the plex state. However, you 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, you can temporarily disable that plex.

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.

Volume Manager utilities use plex states to:

- indicate whether volume contents have been initialized to a known state
- determine if a plex contains a valid copy (mirror) of the volume contents
- track whether a plex was in active use at the time of a system failure
- monitor operations on plexes

This section explains plex states in detail and is intended for administrators who wish to have a detailed knowledge of plex states.

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.

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.

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.

ACTIVE Plex State

A plex can be in the ACTIVE state in two ways:

- when the volume is started and the plex fully participates in normal volume I/O (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 can leave plex contents in an inconsistent state. When a volume is started, Volume Manager does the recovery action to guarantee that the contents of the plexes marked as ACTIVE are made identical.

Note On a system running well, ACTIVE should be the most common state you see for any volume plexes.

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.

OFFLINE Plex State

The `vxmend off task` 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. The plex is not updated until the plex is put online and reattached with the `vxplex att task`. When this occurs, the plex is placed in the STALE state, which causes its contents to be recovered at the next `vxvol start` operation.

TEMP Plex State

Setting a plex to the TEMP state eases 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 sets 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 fails for any reason, a TEMP plex state indicates that the operation is incomplete. A later `vxvol start` dissociates plexes in the TEMP state.

TEMPRM Plex State

A TEMPRM plex state is similar to 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 fails for any reason, the TEMPRM state indicates that the operation did not complete successfully. A later operation dissociates and removes TEMPRM plexes.

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.

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 excluded from the recovery selection process at volume start time.

The Plex State Cycle

The changing of plex states is part normal operations. Changes in plex state indicate abnormalities that Volume Manager must normalize. At system startup, volumes are automatically started and the `vxvol start` task 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.

Plex Kernel State

The *plex kernel state* indicates the accessibility of the plex. The plex kernel state is monitored in the volume driver and allows a plex to have an offline (DISABLED), maintenance (DETACHED), or online (ENABLED) mode of operation.

The following are plex kernel states:

- DISABLED—The plex cannot be accessed.
- DETACHED—A write to the volume is not reflected to the plex. A read request from the volume is not reflected from the plex. Plex operations and ioctl functions are accepted.

- ENABLED—A write request to the volume is reflected to the plex. A read request from the volume is 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.

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, this means that the plexes cannot be guaranteed to be consistent, but are 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, this means that the plexes are being resynchronized through the read-writeback recovery. If the volume is DISABLED, it means that the plexes were being resynchronized through read-writeback when the machine rebooted and therefore still need to be synchronized.
- NEEDSYNC—The volume requires 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 is always set in any case where the volume is marked CLEAN.

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 system 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, 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 requires 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.

Volume Kernel State

The *volume kernel state* indicates the accessibility of the volume. The volume kernel state allows a volume to have an offline (DISABLED), maintenance (DETACHED), or online (ENABLED) mode of operation.

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.

RAID-5 Volume Recovery

This section describes the operation and recovery of RAID-5 volumes. For more information on RAID-5 volumes, see “Understanding Volume Manager” on page 17.

RAID-5 Volume Layout

A RAID-5 volume consists 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 and more efficient recovery

RAID-5 Plexes

RAID-5 volumes keep both the data and parity information in a single RAID-5 plex. A RAID-5 plex consists of subdisks arranged in columns, similar to the striping model. The display below shows a RAID-5 plex and the output of `vxprint` 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
```

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.

Note Your system may use a *device name* that differs from the examples. See “Understanding Volume Manager” on page 17 for more information on device names.

RAID-5 Logs

Each RAID-5 volume has one RAID-5 plex where the data and parity are 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 has 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 concurrent access to the RAID-5 array, the log should be several times the stripe size of the RAID-5 plex.

You can tell the difference between a RAID-5 log plex and a RAID-5 plex of a RAID-5 volume by examining `vxprint` output. The STATE field for a log plex is marked as LOG. The following display 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
```

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 they are associated with a RAID-5 volume and have a layout that is not `RAID`.

Creating RAID-5 Volumes

You can create RAID-5 volumes by using either `vxassist` (recommended) or `vxmake`. Both approaches are described 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.

vxassist and RAID-5 Volumes

You can create a RAID-5 volume by using this `vxassist` command:

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

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

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

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

Initializing RAID-5 Volumes

A RAID-5 volume must be initialized if it was created by `vxmake` and has not yet been initialized or if it has been set to an uninitialized state.

A RAID-5 volume can be initialized by `vxvol` with either of these commands:

```
# 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 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.

Failures and RAID-5 Volumes

Failures are seen in two varieties: *system failures* and *disk failures*. A system failure means that the system has abruptly ceased to operate 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 a system failure (such as a head crash, electronics failure on disk, or disk controller failure).

System Failures

RAID-5 volumes are designed to remain available with a minimum of disk space overhead, if there are disk failures. However, many forms of RAID-5 can have data loss after a system failure. Data loss occurs because a system failure causes the data and parity in the RAID-5 volume to become unsynchronized. Loss of sync occurs because the status of writes that were outstanding at the time of the failure cannot be determined.

If a loss of sync occurs while a RAID-5 volume is being accessed, the volume is described as having *stale parity*. The parity must then be reconstructed by reading all the nonparity 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 causes the data in the volume to be lost. This only applies to RAID-5 volumes *without* log plexes.

[illegible]

Disk Failures

Disk failures can cause the data on a disk to become unavailable. In terms of a RAID-5 volume, this means that a subdisk becomes unavailable.

This can occur due to an uncorrectable I/O error during a write to the disk. The I/O error can cause the subdisk to be detached from the array or a disk being unavailable when the system is booted (for example, from a cabling problem or by 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 is still considered stale and is not used.

If an attempt is made to read data contained on a stale subdisk, the data is reconstructed from data on all other stripe units in the stripe. This operation is called a *reconstructing-read*. This is a more expensive operation than simply reading the data and can result in degraded read performance. 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 from the output of `vxprint`, as shown in 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 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 -
```

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.

A disk containing a RAID-5 log can have 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. In the output of `vxprint -ht`, failure within a RAID-5 log plex is indicated by the plex state being BADLOG. This is shown in the following display, 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
```

RAID-5 Recovery

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

- parity resynchronization
- stale subdisk recovery
- log plex recovery

These types of recovery are described 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
- by calling the `vxrecover` command

For more information on starting RAID-5 volumes, see "Starting RAID-5 Volumes" on page 257.

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 initiate a parity resynchronization.

In the case of a failing RAID-5 log plex, relocation only occurs if the log plex is mirrored; `vxrelocd` then initiates a mirror resynchronization to recreate 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.

Parity Recovery

In most cases, a RAID-5 array does not have stale parity. Stale parity only occurs after all RAID-5 log plexes for the RAID-5 volume have failed, and then only if there is a system failure. Even if a RAID-5 volume has stale parity, it is usually repaired as part of the volume start process.

If a volume without valid RAID-5 logs is started and the process is killed before the volume is resynchronized, the result is 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 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 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
```

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, a synchronization can be manually started by using the `resync` keyword for the `vxvol` command. For example, to resynchronize the RAID-5 volume in the previous display, use this command:

```
# 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. If `-o` *iosize* is not specified, the default maximum I/O size is used. 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 long time. It is possible that the system could be shut down or crash before the operation is completed. In case of a system shutdown, the progress of parity regeneration must be kept across reboots. Otherwise, the process has to start all over again.

To avoid the restart process, parity regeneration is checkpointed. This means 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 the option is not specified, it uses the default checkpoint size.

Because saving the checkpoint offset requires a transaction, making the checkpoint size too small can 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 starts a parity resynchronization at the checkpoint offset.

Subdisk Recovery

Stale subdisk recovery is usually done at volume start time. However, it is possible that the process doing the recovery crashes, or that the volume started with an option to prevent subdisk recovery. It's also possible that the disk on which the subdisk resides was replaced without recovery operations being performed. In any case, a subdisk recovery can be done by using the `recover` keyword of the `vxvol` command. For example, to recover the stale subdisk in the RAID-5 volume shown in Figure 26, "Invalid RAID-5 Volume," on page 258, use this command:

```
# vxvol recover r5vol disk01-00
```

You can have a RAID-5 volume that has multiple stale subdisks to be caught up all at once. Calling `vxvol recover` with only the name of the volume catches multiple stale subdisks:

```
# vxvol recover r5vol
```

Recovering Logs After Failures

RAID-5 log plexes can become detached due to disk failures. 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 27, "Read-Modify-Write," on page 262, use this command:

```
# vxplex att r5vol r5vol-l1
```

Miscellaneous RAID-5 Operations

Many operations exist 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. These command line operations should not be necessary for light usage of the Volume Manager.

Manipulating RAID-5 Logs

RAID-5 logs are represented as plexes of RAID-5 volumes and are manipulated using the `vxplex` command. A RAID-5 log can be added using `vxplex att`:

```
# vxplex att r5vol r5log
```

The attach operation can only proceed if the size of the new log is large enough to hold all of the data on the stripe. If the RAID-5 volume already has logs, the new log length is the minimum of each individual log length. This is because the new log is a mirror of the old logs.

If the RAID-5 volume is not enabled, the new log is marked as BADLOG and is enabled when the volume is started. However, the contents of the log is ignored.

If the RAID-5 volume is enabled and has other enabled RAID-5 logs, the new log has its contents synchronized with the other logs through `ATOMIC_COPY` ioctls.

If the RAID-5 volume currently has no enabled logs, the new log is zeroed before it is enabled.

Log plexes can be removed from a volume by 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 not allowed to continue. The operation must be forced by using the `-o force` option.

Manipulating RAID-5 Subdisks

As with other subdisks, subdisks of the RAID-5 plex of a RAID-5 volume are manipulated using the `vxsd` command. Association is done by 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 the previous display, use this command:

```
# 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 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 by using `VOL_R5_RECOVER` ioctls. Otherwise, it is 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 address space, this places the volume in `DEGRADED` mode. In this case, the `dis` operation prints a warning and must be forced using the `-o force` option. Also, 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 which a RAID-5 volume occupies by using `vxsd mv.` For example, if `disk03` is to be evacuated and `disk22` has enough room by using two portions of its space, you can use this command:

```
# 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 similar.

RAID-5 Subdisk Moves

To do RAID-5 subdisk moves, the current subdisk is removed from the RAID-5 plex and replaced by the new subdisks. The new subdisks are marked as stale and then recovered using `VOL_R5_RECOVER` operations. Recovery is done 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 makes the volume unusable. The RAID-5 volume can also become invalid if the parity of the volume becomes stale.

To avoid these situations, the `vxsd` utility does 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 by 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.

Note RAID-5 subdisk moves are performed the same as other subdisk moves without the penalty of degraded redundancy.

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 can require recovery when it is started, before it can be made available. This section describes actions that can be taken under certain conditions.

Under normal conditions, volumes are started automatically after a reboot and any recovery takes place automatically or is done through 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:

- 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

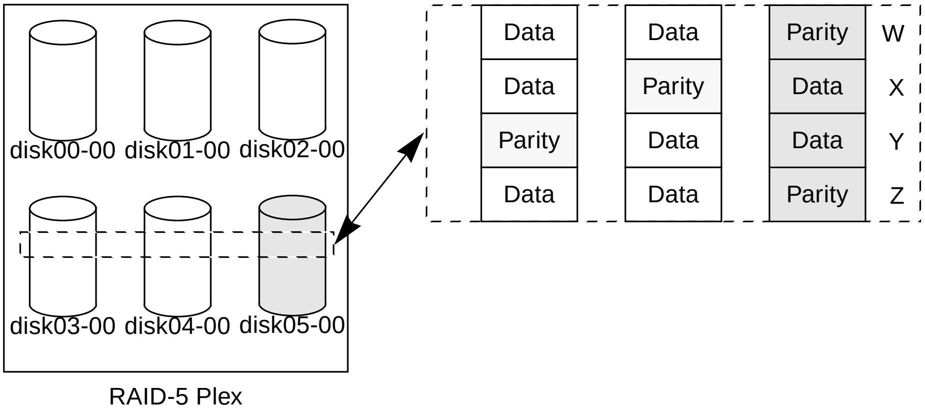
When this occurs, the `vxvol start` command returns this error message:

```
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 possible 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 failed 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 26, ”Invalid RAID-5 Volume,” which shows a RAID-5 volume that has become invalid due to stale parity and a failed subdisk.

Figure 26. 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 display from `vxvol start` is:

```
vxvm:vxvol: ERROR: Volume r5vol is not startable; some subdisks
are unusable and the parity is stale.
```

This situation can be avoided by *always* using two or more RAID-5 log plexes in RAID-5 volumes. RAID-5 log plexes prevent the parity within the volume from becoming stale which prevents this situation (see "System Failures" on page 251 for details).

Forcibly Starting RAID-5 Volumes

You can start a volume even if 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. To prevent this case, always have multiple valid RAID-5 logs associated with the array. However, this may not always be possible.

To start a RAID-5 volume with stale subdisks, you can use the `-f` option with the `vxvol start` command. This causes all stale subdisks to be marked as nonstale. Marking takes place before the `start` operation evaluates the validity of the RAID-5 volume and what is needed to start it. Also, you can mark individual subdisks as nonstale by using the command `vxmend fix unstale` *subdisk*.

Recovery When Starting RAID-5 Volumes

Several operations can be necessary 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 is done to prevent random data from being interpreted as a log entry and corrupting the volume contents. Also, some subdisks may need to be recovered, or the parity may need to be resynchronized (if RAID-5 logs have failed).

The following steps are taken when a RAID-5 volume is started:

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 done 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 makes the volume unusable. This can be overridden by using the `-o unsafe` start option with `vxvol`. If any stale subdisks exist, the RAID-5 volume is unusable.

Caution The `-o unsafe start` option is considered dangerous, as it can make the contents of the volume unusable. It is therefore not recommended.

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 becomes stale, making the volume unusable. If this is undesirable, the volume can be started with the `-o unsafe start` option.

Caution The `-o unsafe` start option is considered dangerous, as it can make the contents of the volume unusable. It is therefore not recommended.

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.

6. If any subdisk recovery fails and there are no valid logs, the volume start is aborted because the subdisk remains stale and a system crash makes the RAID-5 volume unusable. This can also be overridden by using the `-o unsafe` start option.

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 the volume has valid logs, subdisk recovery failures are noted but do not stop the start procedure.

7. When all subdisks have been recovered, the volume is placed in the ENABLED kernel state and marked as ACTIVE. It is now started.

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 changed by using the `vxvol set` command. To change the length of a RAID-5 volume, use this command:

```
# 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. 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, the operation must be forced by using the `-o force usage` option. This is done to prevent removal of space from applications using the volume.

The length of the RAID-5 logs can also be changed by using `vxvol set` with this command:

```
# 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 log length. If increasing the log length makes any of the RAID-5 logs invalid, the operation not allowed. Also, if the volume is not active and is dirty (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).

Writing to RAID-5 Arrays

This section describes the write process for RAID-5 arrays.

Read-Modify-Write

When you write to a RAID-5 array, the following steps can 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.

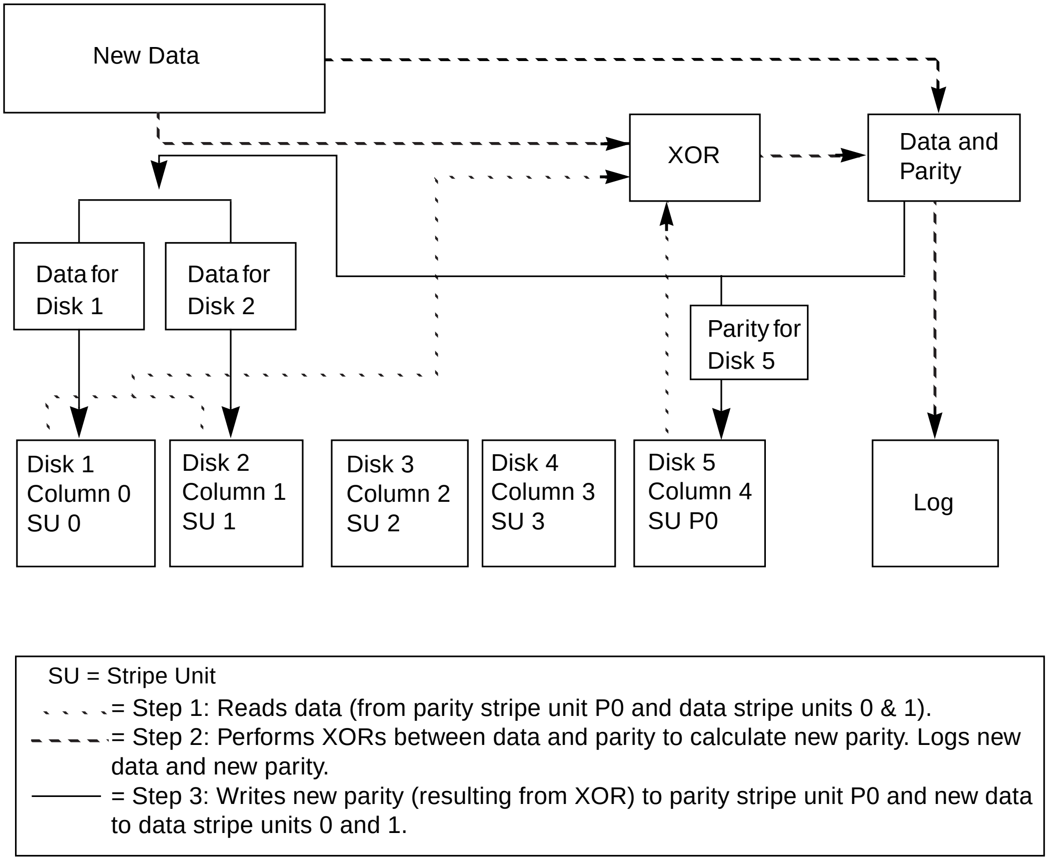
2. Parity is updated to reflect the contents of the new data region. First, the contents of the old data undergo an exclusive OR (XOR) with the parity (logically removing the old data). The new data is then XORed into the parity (logically adding the new data). The new data and new parity are written 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 shown in Figure 27, ”Read-Modify-Write.”

Figure 27. Read-Modify-Write



SU = Stripe Unit

. . . . = Step 1: Reads data (from parity stripe unit P0 and data stripe units 0 & 1).

- - - - = Step 2: Performs XORs between data and parity to calculate new parity. 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 and 1.

Full-Stripe Writes

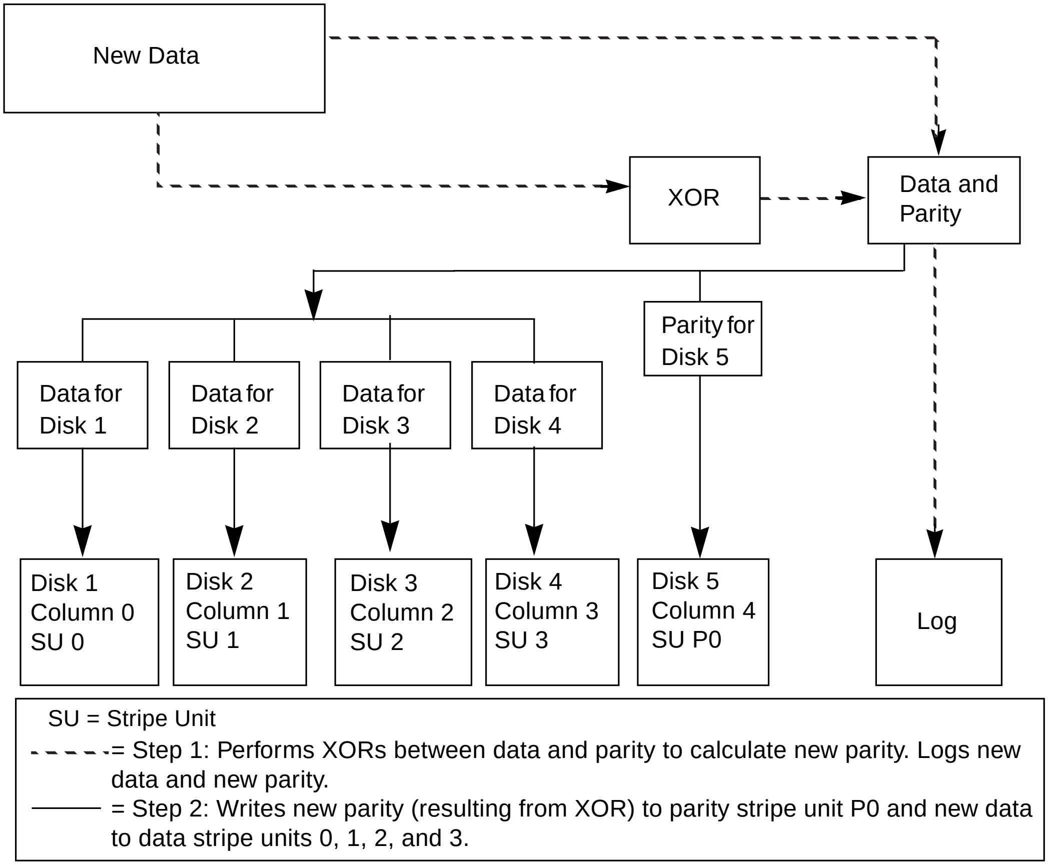
When large writes (writes that cover an entire data stripe) are issued, the read-modify-write procedure can be bypassed in favor of a *full-stripe write*. A full-stripe write is faster than a read-modify-write because it does not require the read process to take place. Eliminating the read cycle reduces 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 28, ”Full-Stripe Write,” shows a full-stripe write.

Figure 28. Full-Stripe Write



SU = Stripe Unit

- - - - -= Step 1: Performs XORs between data and parity to calculate new parity. Logs new data and new parity.

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

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. XORing does not require a read of the parity region and only requires a read of the unaffected data. Unaffected data amounts to less than 50 percent of the stripe units in the stripe.

A reconstruct-write procedure consists of the following steps:

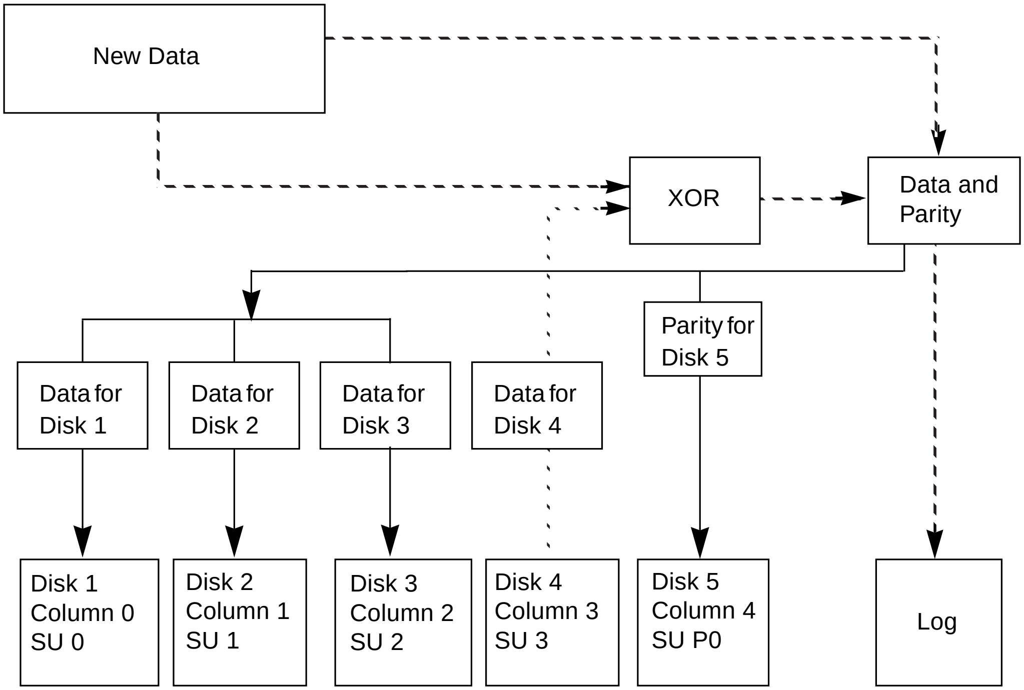
1. Unaffected data is read from the unchanged data stripe unit(s).

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 29, ”Reconstruct-Write,” shows a reconstruct-write. A reconstruct-write is preferable to a read-modify-write in this case because it reads only the necessary data disks, rather than reading the disks and the parity disk.

Figure 29. Reconstruct-Write



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.

VxVM Performance Monitoring 8

Introduction

Logical volume management is a tool that can improve overall system performance. This chapter has performance management and configuration guidelines that can help you to benefit from the advantages provided by Volume Manager. This chapter provides information to establish performance priorities and describes ways to obtain and use appropriate data.

The following topics are covered in this chapter:

- Performance Guidelines
- Performance Monitoring
- Tuning the Volume Manager

Performance Guidelines

Volume Manager provides flexibility in configuring storage to improve system performance. Two basic strategies can be used to optimize performance:

- assign data to physical disks to evenly balance the I/O load among the available disks
- identify the most frequently accessed data and increase access bandwidth to that data by using striping and mirroring

Volume Manager also provides data redundancy (through mirroring and RAID-5) that allows continuous access to data in the event of disk failure.

Data Assignment

When deciding where to locate file systems, a system administrator typically attempts to balance I/O load among available disks. The effectiveness of this approach can 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 disk moves the bottleneck to another disk.

Since Volume Manager can split volumes across multiple disks, 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.

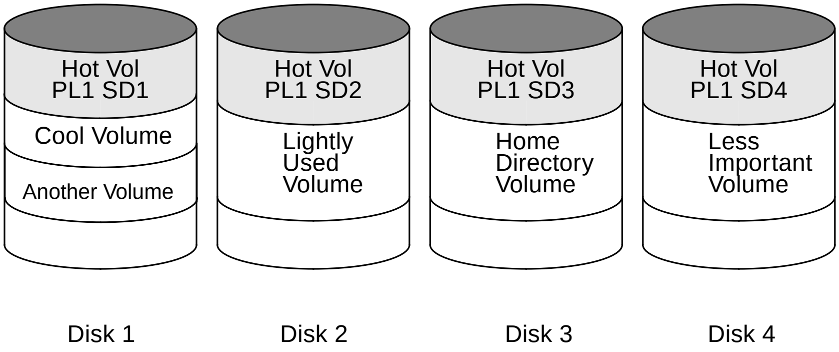
Striping

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

If the most heavily-accessed volumes (containing file systems or databases) can be identified, then performance benefits can be realized. By striping this "high traffic" data across portions of multiple disks, you can increase access bandwidth to this data.

Figure 30, "Use of Striping for Optimal Data Access," 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 30. Use of Striping for Optimal Data Access



Mirroring

Mirroring is a technique for storing multiple copies of data on a system. When properly applied, mirroring can be used to 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.

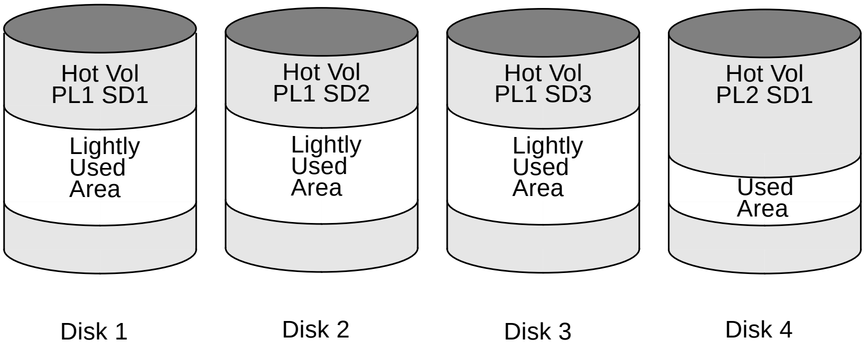
In some cases, mirroring can also be used to improve system performance. Mirroring heavily-accessed data not only protects the data from loss due to disk failure, but can also improve I/O performance. Unlike striping however, performance gained through the use of mirroring depends on the read/write ratio 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, Volume Manager supports these read policies:

- The *round-robin* read policy (`round`), where read requests to the volume are satisfied in a round-robin manner from all plexes in the volume. (This means that the plexes take turns.)
- The *preferred-plex* read policy (`prefer`), where 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. For example, 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 31, "Use of Mirroring and Striping for Improved Performance," 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 31. 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. However, 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.

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.

Striping and Mirroring

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

Striping and mirroring 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. See “*Layered Volumes*” in this manual.

RAID-5

RAID-5 offers many of the advantages of using mirroring and striping together, but RAID-5 requires 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.

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.

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, you can move subdisk locations to balance the disks.

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

Best performance can generally be achieved by striping and mirroring all volumes across a reasonable number of disks and 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), you can place disks in groups of 10 (using disk groups), where each group is used to stripe and mirror a set of volumes. This still provides good performance and eases the task of administration.

Getting Performance Data

Volume Manager 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.

Obtaining I/O Statistics (vxstat)

The `vxstat` utility accesses activity information on volumes, plexes, subdisks, and disks under Volume Manager control. `vxstat` reports statistics that reflect the activity levels of Volume Manager objects since boot time. Statistics for a specific Volume Manager 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 are 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.

Volume Manager records the following I/O statistics:

- a count of operations
- the number of blocks transferred (one operation can involve more than one block)
- the average operation time (which reflects the total time through the Volume Manager interface and is not suitable for comparison against other statistics programs)

Volume Manager records the preceding I/O statistics for logical I/Os. The statistics include 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. Also, 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.

Volume Manager also maintains other statistical data. For each plex, read failures and write failures are maintained. For volumes, corrected read failures and write failures accompany the read failures and write failures.

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

Here 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.

Tracing I/O (vxtrace)

The `vxtrace` command traces operations on volumes. `vxtrace` either prints kernel I/O errors 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 object types or to specified objects or devices. For additional information, refer to the `vxtrace`(1M) manual page.

Using Performance Data

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

Using I/O Statistics

Examination of the I/O statistics can suggest reconfiguration. There are two primary statistics: volume I/O activity and disk I/O activity.

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

After clearing the statistics, allow the system to run during typical system activity. To measure the effect of a particular application or workload, the system should be run on that particular application or workload. 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. Here is a typical display of volume statistics:

```
                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, use the command `vxstat -d`. Here is a typical display of disk statistics:

```
                 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
```

You may want 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 this command:

```
# vxprint -tvh archive
```

Here is an example display:

```
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 device
```

Note Your system may use a *device name* that differs from the examples. See "*Understanding Volume Manager*," in this manual for more information on device names.

The associated subdisks list indicates that the `archive` volume is on disk `disk03`. To move the volume off `disk03`, use the command:

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

where *dest_disk* is the disk to which you want to move the volume. It is not necessary to specify a *dest_disk*. If you do not, the volume is moved to an available disk with enough space to contain the volume.

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

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

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

Note The graphical user interface provides an easy way to move pieces of volumes between disks and may be preferable to using the command-line.

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

If there is one volume that is particularly busy (especially if it has unusually large average read or write times), stripe the volume (or split 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 these commands:

```
# 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 busy (or have particularly long read or write times), you may want to reconfigure some volumes. If there are two relatively busy volumes on a disk, move them closer together to reduce seek times on the disk. If there are too many relatively busy volumes on one disk, move them to a disk that is less busy.

Use I/O tracing (or subdisk statistics) to determine whether volumes have excessive activity in particular regions of the volume. If the active regions can be identified, split the subdisks in the volume and 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 results 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 requires 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 results 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.

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 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.

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).

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. 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. On some systems, 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.

Tunables

On some systems, the `idtune` command should be used to modify tunables. Refer to the `idtune`(1M) manual page for details.

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

For example, to change the default value of a tunable called `vol_tuneme` to a value of 5000, the following line should be inserted 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 following sections that describe specific tunables.

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.

voliomem_maxpool_sz

The purpose of this tunable is to prevent one I/O from using all the memory in the system.

VxVM allocates two pools of voliomem_maxpool_sz, one for RAID-5 and one for mirrored volumes.

When a write for a RAID-5 volume comes in and is greater than volio_maxpoll_sz/10, it is broken up and performed in chunks of volio_maxpoll_sz/10.

When a write for a mirrored volume comes in and is greater than volio_maxpoll_sz/2, it is broken up and performed in chunks of volio_maxpoll_sz/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.

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 allow 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).

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).

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 cause difficulties if the size of an I/O causes the process to take more memory or kernel mapping space than exists and thus deadlock. The maximum limit for `vol_maxio` is 20% of the smaller of physical memory or kernel virtual memory. It is inadvisable to go over this limit since deadlock is likely to occur.

If you have stripes larger than `vol_maxio`, full stripe I/O's are broken up which prevents full-stripe read/writes. This throttles the volume I/O throughput for sequential I/O or larger I/O.

This tunable should be set, as minimum, to the size of your largest stripe. This tunable guideline applies to both Raid0 striping and Raid5 striping.

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.

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.

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.

voldrl_max_dirty

Some volumes, such as those used for Oracle replay logs, are written sequentially and do not benefit from this lazy cleaning of the DRL bits. For these volumes, *sequential DRL* can be used to further restrict the number of dirty bits and speed up recovery. The number of dirty bits allowed for sequential DRL will be restricted by the tunable voldrl_max_dirty. Using sequential DRL on volumes that are written sequentially may severely impact I/O throughput.

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.

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.

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.

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).

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. Care should be taken not to increase this value above the value for the `voliot_iobuf_limit` tunable value.

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.

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.

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.

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 reduces the overhead of checkpointing on recovery operations at the expense of additional recovery following a system failure during a recovery.

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.

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 Volume Manager to keep hold of a larger amount of memory.

The default size for this tunable is 64K.

Tuning for Large Systems

On smaller systems (less than 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 allow 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 either has to be split into multiple disk groups, or the private regions have to be enlarged. This involves re-initializing each disk in the disk group (and can 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 so the disk group can be physically moved as a unit between systems.

The Number of Configuration Copies for a Disk Group

Selection of the number of configuration copies for a disk group is based on the trade-off between redundancy and performance. As a general 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 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 sufficient from the redundancy perspective, but can lead to large numbers of configuration copies under some circumstances.

If this is the case, it is recommended to limit the number of configuration copies to a minimum of 4. The location of the copies is 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). Also, you can change copies of an existing group by using 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, use this command:

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

Glossary

Active/Active disk arrays

This type of multipathed disk array allows you to access a disk in the disk array through all the paths to the disk simultaneously, without any performance degradation.

Active/Passive disk arrays

This type of multipathed disk array allows one path to a disk to be designated as primary and used to access the disk at any time. Using a path other than the designated active path results in severe performance degradation in some disk arrays. See “*path*”, “*primary path*”, “*secondary path*”.

associate

The process of establishing a relationship between Volume Manager objects; for example, a subdisk that has been created and defined as having a starting point within a plex is referred to as being associated with that plex.

associated plex

A plex associated with a volume.

associated subdisk

A subdisk associated with a plex.

atomic operation

An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.

attached

A state in which a VxVM object is both associated with another object and enabled for use.

block

The minimum unit of data transfer to a disk or array.

boot disk

A disk used for booting purposes. This disk may be under VxVM control.

clean node shutdown

The ability of a node to leave the cluster gracefully when all access to shared volumes has ceased.

cluster

A set of hosts that share a set of disks.

cluster manager

An externally-provided daemon that runs on each node in a cluster. The cluster managers on each node communicate with each other and inform VxVM of changes in cluster membership.

cluster-shareable disk group

A disk group in which the disks are shared by multiple hosts (also referred to as a *shared disk group*).

column

A set of one or more subdisks within a striped plex. Striping is achieved by allocating data alternately and evenly across the columns within a plex.

concatenation

A layout style characterized by subdisks that are arranged sequentially and contiguously.

configuration database

A set of records containing detailed information on existing Volume Manager objects (such as disk and volume attributes). A single copy of a configuration database is called a configuration copy.

data stripe

This represents the usable data portion of a stripe and is equal to the stripe minus the parity region.

detached

A state in which a VxVM object is associated with another object, but not enabled for use.

device name

The device name or address used to access a physical disk, such as `c0t0d0`. The `c#t#d#s#` syntax identifies the controller, target address, disk, and area (on some systems, partition.

Dirty Region Logging

The procedure by which the Volume Manager monitors and logs modifications to a plex. A bitmap of changed regions is kept in an associated subdisk called a *log subdisk*.

disabled path

A path to a disk that is not available for I/O. A path can be *disabled* due to real hardware failures or if the user has used the `vxdmpadm disable` command on that controller.

disk

A collection of read/write data blocks that are indexed and can be accessed fairly quickly. Each disk has a universally unique identifier.

disk access name

The name used to access a physical disk, such as `c0t0d0`. The `c#t#d#s#` syntax identifies the controller, target address, disk, and partition. The term *device name* can also be used to refer to the disk access name.

disk access records

Configuration records used to specify the access path to particular disks. Each disk access record contains a name, a type, and possibly some type-specific information, which is used by the Volume Manager in deciding how to access and manipulate the disk that is defined by the disk access record.

disk array

A collection of disks logically arranged into an object. Arrays tend to provide benefits such as redundancy or improved performance.

disk array serial number

This is the serial number of the disk array. It is usually printed on the disk array cabinet or can be obtained by issuing a vendor specific SCSI command to the disks on the disk array. This number is used by the DMP subsystem to uniquely identify a disk array.

disk controller

The controller (HBA) connected to the host *or* the disk array that

is represented as the parent node of the disk by the Operating System, is called

the disk controller by the multipathing subsystem of Volume Manager.

For example, if a disk is represented by the device name:

```
/devices/sbus@1f,0/QLGC,isp@2,10000/sd@8,0:c
```

then the disk controller for the disk `sd@8,0`:c is:

```
QLGC,isp@2,10000
```

This controller (HBA) is connected to the host.

disk group

A collection of disks that share a common configuration. A disk group configuration is a set of records containing detailed information on existing Volume Manager objects (such as disk and volume attributes) and their relationships. Each disk group has an administrator-assigned name and an internally defined unique ID. The root disk group (`rootdg`) is a special private disk group that always exists.

disk group ID

A unique identifier used to identify a disk group.

disk ID

A universally unique identifier that is given to each disk and can be used to identify the disk, even if it is moved.

disk media name

A logical or administrative name chosen for the disk, such as `disk03`. The term *disk name* is also used to refer to the disk media name.

disk media record

A configuration record that identifies a particular disk, by disk ID, and gives that disk a logical (or administrative) name.

dissociate

The process by which any link that exists between two Volume Manager objects is removed. For example, dissociating a subdisk from a plex removes the subdisk from the plex and adds the subdisk to the free space pool.

dissociated plex

A plex dissociated from a volume.

dissociated subdisk

A subdisk dissociated from a plex.

distributed lock manager

A lock manager that runs on different systems and ensures consistent access to distributed resources.

enabled path

A path to a disk that is available for I/O.

encapsulation

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. Encapsulation is not applicable on some systems.

file system

A collection of files organized together into a structure. The UNIX file system is a hierarchical structure consisting of directories and files.

free space

An area of a disk under VxVM control that is not allocated to any subdisk or reserved for use by any other Volume Manager object.

free subdisk

A subdisk that is not associated with any plex and has an empty `putil[0]` field.

hostid

A string that identifies a host to the Volume Manager. The *hostid* for a host is stored in its `volboot` file, and is used in defining ownership of disks and disk groups.

hot-relocation

A technique of automatically restoring redundancy and access to mirrored and RAID-5 volumes when a disk fails. This is done by relocating the affected subdisks to disks designated as spares and/or free space in the same disk group.

initiating node

The node on which the system administrator is running a utility that requests a change to Volume Manager objects. This node initiates a volume reconfiguration.

log plex

A plex used to store a RAID-5 log. The term *log plex* may also be used to refer to a Dirty Region Logging plex.

log subdisk

A subdisk that is used to store a dirty region log. See *Dirty Region Logging*.

master node

A node that is designated by the software as the “master” node. Any node is capable of being the master node. The master node coordinates certain Volume Manager operations.

mastering node

The node to which a disk is attached. This is also known as a *disk owner.*

mirror

A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms *mirror* and *plex* can be used synonymously.

mirroring

A layout technique that mirrors the contents of a volume onto multiple plexes. Each plex duplicates the data stored on the volume, but the plexes themselves may have different layouts.

multipathing

Where there are multiple physical access paths to a disk connected to a system, the disk is called multipathed. Any software residing on the host, (e.g., the DMP driver) that hides this fact from the user is said to provide multipathing functionality.

node

One of the hosts in a cluster.

node abort

A situation where a node leaves a cluster (on an emergency basis) without attempting to stop ongoing operations.

node join

The process through which a node joins a cluster and gains access to shared disks.

object

An entity that is defined to and recognized internally by the Volume Manager. The VxVM objects are: volume, plex, subdisk, disk, and disk group. There are actually two types of disk objects—one for the physical aspect of the disk and the other for the logical aspect.

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 recreated from the remaining data and the parity.

parity stripe unit

A RAID-5 volume storage region that contains parity information. The data contained in the parity stripe unit can be used to help reconstruct regions of a RAID-5 volume that are missing because of I/O or disk failures.

partition

The standard division of a physical disk device, as supported directly by the operating system and disk drives.

path

When a disk is connected to a host, the path to the disk consists of the HBA (Host Bus Adapter) on the host, the SCSI or fibre cable connector and the controller on the disk or disk array. These components constitute a path to a disk. A failure on any of these results in DMP trying to shift all I/Os for that disk onto the remaining(alternate) paths.

persistent state logging

A logging type that ensures that only active mirrors are used for recovery purposes and prevents failed mirrors from being selected for recovery. This is also known as *kernel logging*.

physical disk

The underlying storage device, which may or may not be under Volume Manager control.

plex

A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each plex is one copy of the volume with which the plex is associated. The terms *mirror* and *plex* can be used synonymously.

primary path

In Active/Passive type disk arrays, a disk can be bound to one particular controller on the disk array or owned by a controller. The disk can then be accessed using the path through this particular controller. See “*path*”, “*secondary path*”.

private disk group

A disk group in which the disks are accessed by only one specific host.

private region

A region of a physical disk used to store private, structured Volume Manager information. The *private region* contains a disk header, a table of contents, and a configuration database. The table of contents maps the contents of the disk. The disk header contains a disk ID. All data in the private region is duplicated for extra reliability.

public region

A region of a physical disk managed by the Volume Manager that contains available space and is used for allocating subdisks.

RAID

A Redundant Array of Independent Disks (RAID) is a disk array set up with part of the combined storage capacity used for storing duplicate information about the data stored in that array. This makes it possible to regenerate the data if a disk failure occurs.

read-writeback mode

A recovery mode in which each read operation recovers plex consistency for the region covered by the read. Plex consistency is recovered by reading data from blocks of one plex and writing the data to all other writable plexes.

root configuration

The configuration database for the root disk group. This is special in that it always contains records for other disk groups, which are used for backup purposes only. It also contains disk records that define all disk devices on the system.

root disk

The disk containing the root file system. This disk may be under VxVM control.

root disk group

A special private disk group that always exists on the system. The root disk group is named `rootdg`.

root file system

The initial file system mounted as part of the UNIX kernel startup sequence.

root partition

The disk region on which the root file system resides.

root volume

The VxVM volume that contains the root file system, if such a volume is designated by the system configuration.

rootability

The ability to place the `root` file system and the `swap` device under Volume Manager control. The resulting volumes can then be mirrored to provide redundancy and allow recovery in the event of disk failure.

secondary path

In Active/Passive type disk arrays, the paths to a disk other than the primary path are called secondary paths. A disk is supposed to be accessed only through the primary path until it fails, after which ownership of the disk is transferred to one of the secondary paths. See “path”, “primary path”.

sector

A unit of size, which can vary between systems. A sector is commonly 512 bytes.

shared disk group

A disk group in which the disks are shared by multiple hosts (also referred to as a *cluster-shareable disk group*).

shared volume

A volume that belongs to a shared disk group and is open on more than one node at the same time.

shared VM disk

A VM disk that belongs to a shared disk group.

slave node

A node that is not designated as a master node.

slice

The standard division of a logical disk device. The terms *partition* and *slice* are sometimes used synonymously.

spanning

A layout technique that permits a volume (and its file system or database) too large to fit on a single disk to span across multiple physical disks.

sparse plex

A plex that is not as long as the volume or that has holes (regions of the plex that don't have a backing subdisk).

stripe

A set of stripe units that occupy the same positions across a series of columns.

stripe size

The sum of the stripe unit sizes comprising a single stripe across all columns being striped.

stripe unit

Equally-sized areas that are allocated alternately on the subdisks (within columns) of each striped plex. In an array, this is a set of logically contiguous blocks that exist on each disk before allocations are made from the next disk in the array. A *stripe unit* may also be referred to as a *stripe element*.

stripe unit size

The size of each stripe unit. The default stripe unit size is 32 sectors (16K). A *stripe unit size* has also historically been referred to as a *stripe width*.

striping

A layout technique that spreads data across several physical disks using stripes. The data is allocated alternately to the stripes within the subdisks of each plex.

subdisk

A consecutive set of contiguous disk blocks that form a logical disk segment. Subdisks can be associated with plexes to form volumes.

swap area

A disk region used to hold copies of memory pages swapped out by the system pager process.

swap volume

A VxVM volume that is configured for use as a swap area.

transaction

A set of configuration changes that succeed or fail as a group, rather than individually. Transactions are used internally to maintain consistent configurations.

`volboot` file

A small file that is used to locate copies of the root configuration. The file may list disks that contain configuration copies in standard locations, and can also contain direct pointers to configuration copy locations. `volboot` is stored in a system-dependent location.

VM disk

A disk that is both under Volume Manager control and assigned to a disk group. VM disks are sometimes referred to as *Volume Manager disks* or simply *disks*. In the graphical user interface, VM disks are represented iconically as cylinders labeled `D`.

volume

A virtual disk, representing an addressable range of disk blocks used by applications such as file systems or databases. A volume is a collection of from one to 32 plexes.

volume configuration device

The volume configuration device (`/dev/vx/config`) is the interface through which all configuration changes to the volume device driver are performed.

volume device driver

The driver that forms the virtual disk drive between the application and the physical device driver level. The volume device driver is accessed through a virtual disk device node whose character device nodes appear in `/dev/vx/rdsk`, and whose block device nodes appear in `/dev/vx/dsk.`

volume event log

The volume event log device (`/dev/vx/event`) is the interface through which volume driver events are reported to the utilities.

vxconfigd

The Volume Manager configuration daemon, which is responsible for making changes to the VxVM configuration. This daemon must be running before VxVM operations can be performed.