Veritas™ Dynamic Multi-Pathing Administrator's Guide

AIX

5.1 Service Pack 1

Veritas™ Dynamic Multi-Pathing Administrator's Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Product version: 5.1 SP1

Document version: 5.1SP1.1

Legal Notice

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043

http://www.symantec.com

Technical Support

Symantec Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's support offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and/or Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers software upgrades
- Global support purchased on a regional business hours or 24 hours a day, 7 days a week basis
- Premium service offerings that include Account Management Services

For information about Symantec's support offerings, you can visit our Web site at the following URL:

www.symantec.com/business/support/index.jsp

All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policy.

Contacting Technical Support

Customers with a current support agreement may access Technical Support information at the following URL:

www.symantec.com/business/support/contact_techsupp_static.jsp

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level

- Hardware information
- Available memory, disk space, and NIC information
- Operating system
- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
  - Error messages and log files
  - Troubleshooting that was performed before contacting Symantec
  - Recent software configuration changes and network changes

Licensing and registration

If your Symantec product requires registration or a license key, access our technical support Web page at the following URL:

www.symantec.com/business/support/

Customer service

Customer service information is available at the following URL:

www.symantec.com/business/support/

Customer Service is available to assist with non-technical questions, such as the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and support contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Documentation

Product guides are available on the media in PDF format. Make sure that you are using the current version of the documentation.The document version appears on page 2 of each guide. The latest product documentation is available on the Symantec website.

http://www.symantec.com/business/support/overview.jsp?pid=15107

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions. Include the title and document version (located on the second page), and chapter and section titles of the text on which you are reporting. Send feedback to:

docs@symantec.com

About Symantec Connect

Symantec Connect is the peer-to-peer technical community site for Symantec's enterprise customers. Participants can connect and share information with other product users, including creating forum posts, articles, videos, downloads, blogs and suggesting ideas, as well as interact with Symantec product teams and Technical Support. Content is rated by the community, and members receive reward points for their contributions.

http://www.symantec.com/connect/storage-management

Support agreement resources

If you want to contact Symantec regarding an existing support agreement, please contact the support agreement administration team for your region as follows:

Asia-Pacific and Japan	customercare_apac@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Contents

Chapter 1

Understanding DMP

This chapter includes the following topics:

About Veritas Dynamic Multi-Pathing

Veritas Dynamic Multi-Pathing (DMP) provides multi-pathing functionality for the operating system native devices configured on the system. DMP creates DMP metadevices (also known as DMP nodes) to represent all the device paths to the same physical LUN.

In previous Veritas releases, DMP was only available as a feature of Veritas Volume Manager (VxVM). DMP supported VxVM volumes on DMP metadevices, and Veritas File System (VxFS) file systems on those volumes.

This release extends DMP metadevices to support OS native logical volume managers (LVM). You can create LVM volumes and volume groups on DMP metadevices.

DMP supports LVM volume devices that are used as the paging devices.

In this release, Veritas Dynamic Multi-Pathing does not support Veritas File System (VxFS) on DMP devices.

Veritas Volume Manager (VxVM) volumes and disk groups can co-exist with LVM volumes and volume groups, but each device can only support one of the types. If a disk has a VxVM label, then the disk is not available to LVM. Similarly, if a disk is in use by LVM, then the disk is not available to VxVM.

How DMP works

Veritas Dynamic Multi-Pathing (DMP) provides greater availability, reliability, and performance by using path failover and load balancing. This feature is available for multiported disk arrays from various vendors.

Multiported disk arrays can be connected to host systems through multiple paths. To detect the various paths to a disk, DMP uses a mechanism that is specific to each supported array. DMP can also differentiate between different enclosures of a supported array that are connected to the same host system.

See "Discovering and configuring newly added disk devices" on page 133.

The multi-pathing policy that is used by DMP depends on the characteristics of the disk array.

DMP supports the following standard array types:

Active/Active (A/A)	Allows several paths to be used concurrently for I/O. Such arrays allow DMP to provide greater I/O throughput by balancing the I/O load uniformly across the multiple paths to the LUNs. In the event that one path fails, DMP automatically routes I/O over the other available paths.
Asymmetric Active/Active (A/A-A)	A/A-A or Asymmetric Active/Active arrays can be accessed through secondary storage paths with little performance degradation. Usually an A/A-A array behaves like an A/P array rather than an A/A array. However, during failover, an A/A-A array behaves like an A/A array. An ALUA array behaves like an A/A-A array.

Active/Passive (A/P)	Allows access to its LUNs (logical units; real disks or virtual disks created using hardware) via the primary (active) path on a single controller (also known as an access port or a storage processor) during normal operation. In implicit failover mode (or autotrespass mode), an A/P array automatically fails over by scheduling I/O to the secondary (passive) path on a separate controller if the primary path fails. This passive port is not used for I/O until the active port fails. In A/P arrays, path failover can occur for a single LUN if I/O fails on the primary path. This policy supports concurrent I/O and load balancing by having multiple primary paths into a controller. This functionality is provided by a controller with multiple ports, or by the insertion of a SAN switch between an array and a controller. Failover to the secondary (passive) path occurs only if all the active primary paths fail.
Active/Passive in explicit failover mode or non-autotrespass mode (A/P-F)	The appropriate command must be issued to the array to make the LUNs fail over to the secondary path. This policy supports concurrent I/O and load balancing by having multiple primary paths into a controller. This functionality is provided by a controller with multiple ports, or by the insertion of a SAN switch between an array and a controller. Failover to the secondary (passive) path occurs only if all the active primary paths fail.

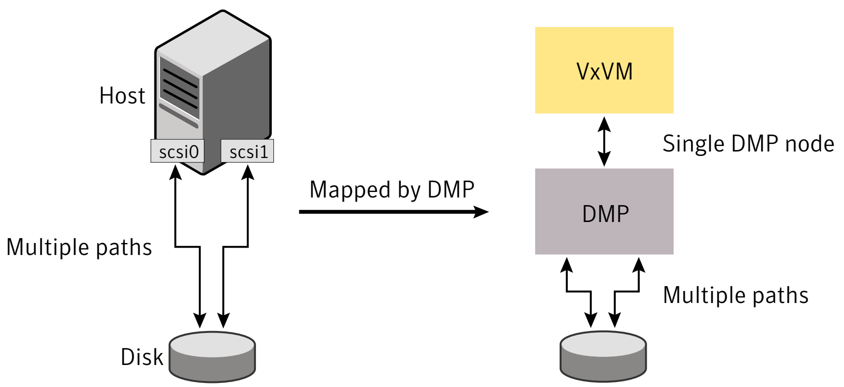
Active/Passive with LUN group failover (A/P-G)	For Active/Passive arrays with LUN group failover (A/PG arrays), a group of LUNs that are connected through a controller is treated as a single failover entity. Unlike A/P arrays, failover occurs at the controller level, and not for individual LUNs. The primary controller and the secondary controller are each connected to a separate group of LUNs. If a single LUN in the primary controller's LUN group fails, all LUNs in that group fail over to the secondary controller. This policy supports concurrent I/O and load balancing by having multiple primary paths into a controller. This functionality is provided by a controller with multiple ports, or by the insertion of a SAN switch between an array and a controller. Failover to the secondary (passive) path occurs only if all the active primary paths fail.

An array policy module (APM) may define array types to DMP in addition to the standard types for the arrays that it supports.

VxVM uses DMP metanodes (DMP nodes) to access disk devices connected to the system. For each disk in a supported array, DMP maps one node to the set of paths that are connected to the disk. Additionally, DMP associates the appropriate multi-pathing policy for the disk array with the node. For disks in an unsupported array, DMP maps a separate node to each path that is connected to a disk. The raw and block devices for the nodes are created in the directories `/dev/vx/rdmp` and `/dev/vx/dmp` respectively.

Figure 1-1 shows how DMP sets up a node for a disk in a supported disk array.

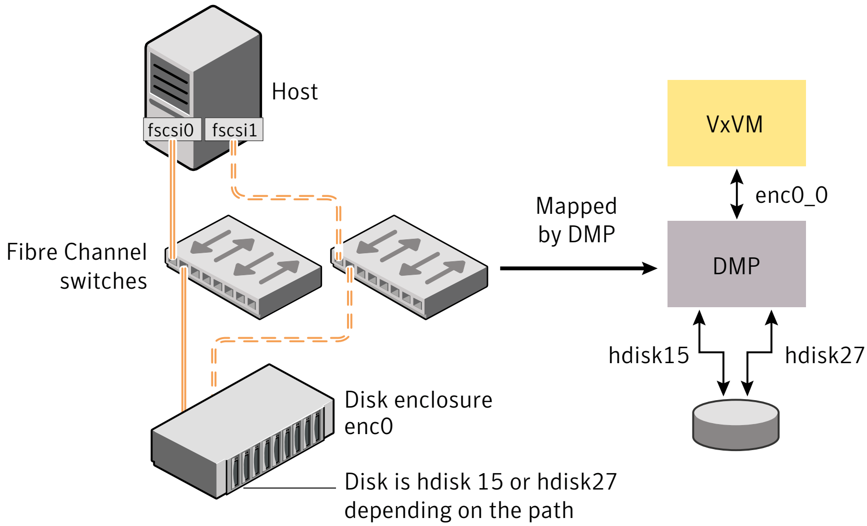
Figure 1-1 How DMP represents multiple physical paths to a disk as one node



VxVM implements a disk device naming scheme that allows you to recognize to which array a disk belongs.

Figure 1-2 shows an example where two paths, `hdisk15` and `hdisk27`, exist to a single disk in the enclosure, but VxVM uses the single DMP node, `enc0_0`, to access it.

Figure 1-2 Example of multi-pathing for a disk enclosure in a SAN environment



See "About enclosure-based naming" on page 25.

See "Discovering and configuring newly added disk devices" on page 133.

How DMP monitors I/O on paths

In older releases of VxVM, DMP had one kernel daemon (`errord`) that performed error processing, and another (`restored`) that performed path restoration activities.

From release 5.0, DMP maintains a pool of kernel threads that are used to perform such tasks as error processing, path restoration, statistics collection, and SCSI request callbacks. The `vxdmpadm stat` command can be used to provide information about the threads. The names `errord` and `restored` have been retained for backward compatibility.

One kernel thread responds to I/O failures on a path by initiating a probe of the host bus adapter (HBA) that corresponds to the path. Another thread then takes the appropriate action according to the response from the HBA. The action taken can be to retry the I/O request on the path, or to fail the path and reschedule the I/O on an alternate path.

The restore kernel task is woken periodically (typically every 5 minutes) to check the health of the paths, and to resume I/O on paths that have been restored. As some paths may suffer from intermittent failure, I/O is only resumed on a path if the path has remained healthy for a given period of time (by default, 5 minutes). DMP can be configured with different policies for checking the paths.

See "Configuring DMP path restoration policies" on page 129.

The statistics-gathering task records the start and end time of each I/O request, and the number of I/O failures and retries on each path. DMP can be configured to use this information to prevent the SCSI driver being flooded by I/O requests. This feature is known as I/O throttling.

If an I/O request relates to a mirrored volume, VxVM specifies the FAILFAST flag. In such cases, DMP does not retry failed I/O requests on the path, and instead marks the disks on that path as having failed.

See "Path failover mechanism" on page 18.

See "I/O throttling" on page 19.

Path failover mechanism

DMP enhances system reliability when used with multiported disk arrays. In the event of the loss of a path to a disk array, DMP automatically selects the next available path for I/O requests without intervention from the administrator.

DMP is also informed when a connection is repaired or restored, and when you add or remove devices after the system has been fully booted (provided that the operating system recognizes the devices correctly).

If required, the response of DMP to I/O failure on a path can be tuned for the paths to individual arrays. DMP can be configured to time out an I/O request either after a given period of time has elapsed without the request succeeding, or after a given number of retries on a path have failed.

See "Configuring the response to I/O failures" on page 124.

Subpaths Failover Group (SFG)

An SFG represents a group of paths which could fail and restore together. When an I/O error is encountered on a path in an SFG group, DMP does proactive path probing on the other paths of that SFG as well. This behavior adds greatly to the performance of path failover thus improving IO performance. Currently the criteria followed by DMP to form the subpath failover groups is to bundle the paths with the same endpoints from the host to the array into one logical storage failover group.

See "Configuring Subpaths Failover Groups (SFG)" on page 127.

Low Impact Path Probing (LIPP)

The restore daemon in DMP keeps probing the LUN paths periodically. This behavior helps DMP to keep the path states up-to-date even though IO activity is not there on the paths. Low Impact Path Probing adds logic to the restore daemon to optimize the number of the probes performed while the path status is being updated by the restore daemon. This optimization is achieved with the help of the logical subpaths failover groups. With LIPP logic in place, DMP probes only limited number of paths within an SFG, instead of probing all the paths in an SFG. Based on these probe results, DMP determines the states of all the paths in that SFG.

See "Configuring Low Impact Path Probing" on page 127.

I/O throttling

If I/O throttling is enabled, and the number of outstanding I/O requests builds up on a path that has become less responsive, DMP can be configured to prevent new I/O requests being sent on the path either when the number of outstanding I/O requests has reached a given value, or a given time has elapsed since the last successful I/O request on the path. While throttling is applied to a path, the new I/O requests on that path are scheduled on other available paths. The throttling is removed from the path if the HBA reports no error on the path, or if an outstanding I/O request on the path succeeds.

See "Configuring the I/O throttling mechanism" on page 126.

Load balancing

By default, the DMP uses the Minimum Queue policy for load balancing across paths for Active/Active (A/A), Active/Passive (A/P), Active/Passive with explicit failover (A/P-F) and Active/Passive with group failover (A/P-G) disk arrays. Load balancing maximizes I/O throughput by using the total bandwidth of all available paths. I/O is sent down the path which has the minimum outstanding I/Os.

For A/P disk arrays, I/O is sent down the primary paths. If the primary paths fail, I/O is switched over to the available secondary paths. As the continuous transfer of ownership of LUNs from one controller to another results in severe I/O slowdown, load balancing across primary and secondary paths is not performed for A/P disk arrays unless they support concurrent I/O.

For A/P, A/P-F and A/P-G arrays, load balancing is performed across all the currently active paths as is done for A/A arrays.

You can use the `vxdmpadm` command to change the I/O policy for the paths to an enclosure or disk array.

See "Specifying the I/O policy" on page 115.

Using DMP with LVM boot disks

The Logical Volume Manager (LVM) in AIX is incapable of switching between multiple paths that may exist to the boot disk. If the path that LVM selects becomes unavailable at boot time, the `root` file system is disabled, and the boot fails. DMP can be configured to overcome this problem by ensuring that an alternate path is available at boot time.

Support for LVM bootability over DMP is enabled by running the following command:

```
# /usr/sbin/vxdmpadm native enable vgname=rootvg
```

Individual DMP nodes or subpaths can be added or removed from the rootvg. The following command needs to be executed after adding or removing the DMP node or subpaths:

```
# /usr/sbin/vxdmpadm native enable vgname=rootvg
```

Support for LVM bootability over DMP is disabled by running the following command:

```
# /usr/sbin/vxdmpadm native disable vgname=rootvg
```

LVM bootability over DMP can be verified as being enabled on a system using the following command:

```
# /usr/sbin/vxdmpadm native list vgname=rootvg
```

See the `vxdmpadm`(1M) manual page.

Disabling MPIO

The Multiple Path I/O (MPIO) feature was introduced in AIX 5.2 to manage disks and LUNs with multiple paths. By default, MPIO is enabled on all disks and LUNs that have this capability, which prevents DMP or other third-party multi-pathing drivers (such as EMC PowerPath) from managing the paths to such devices.

To allow DMP or a third-party multi-pathing driver to manage multi-pathing instead of MPIO, you must install suitable Object Data Manager (ODM) definitions for the devices on the host. Without these ODM definitions, MPIO consolidates the paths, and DMP can only see a single path to a given device.

There are several reasons why you might want to configure DMP to manage multi-pathing instead of MPIO:

- Using DMP can enhance array performance if an ODM defines properties such as queue depth, queue type and timeout for the devices.
- The I/O fencing features of the Storage Foundation HA or Storage Foundation Real Application Cluster software do not work with MPIO devices.

Contact the array vendor to obtain ODM definitions for the array type and the version of AIX on your system. The ODM definition should permit either DMP or the array vendor's multi-pathing driver to discover the devices in the supported array.

Some array vendors do not distribute ODM Pre-defines for their arrays for AIX. In this case, you can use the devices as hdisk devices, as long as MPIO does not claim these LUNs.

Having obtained the filesets that contain the new ODM definitions, use the following procedure to configure DMP or an array vendor's multi-pathing driver in place of MPIO.

To disable MPIO

1. Unmount any file systems and stop all applications such as databases that are configured on VxVM volumes.
2. Stop all I/O to the VxVM volumes by entering the following command for each disk group:

   ```
   # vxvol -g diskgroup stopall
   ```

3. Use the `vxprint` command to verify that no volumes remain open:

   ```
   # vxprint -Aht -e v_open
   ```

4. Deport each disk group in turn:

   ```
   # vxdg deport diskgroup
   ```

5. Use the following command to remove each `hdisk` device that MPIO has configured to the arrays:

   ```
   # rmdev -dl hdisk_device
   ```

   Alternatively, use the `smitty rmdev` command.

6 Use the `installp` command to install the replacement ODM filesets:

```
# installp -agXd ODM_fileset ...
```

Alternately, you can use the smitty `installp` command.

7 Reboot the system so that the new ODM definitions are used to perform device discovery.

8 Use the `vxdmpadm` command to check that DMP now has access to all the paths to the devices. The following command displays a list of HBA controllers that are configured on a system:

```
# vxdmpadm listctlr all
```

The next command displays information about all the paths that are connected to a particular HBA controller (for example, `fscsi2`):

```
# vxdmpadm getsubpaths ctlr=controller
```

DMP in a clustered environment

Note: You need an additional license to use the cluster feature of VxVM.

Clustering is only supported for VxVM.

In a clustered environment where Active/Passive type disk arrays are shared by multiple hosts, all nodes in the cluster must access the disk via the same physical storage controller port. Accessing a disk via multiple paths simultaneously can severely degrade I/O performance (sometimes referred to as the ping-pong effect). Path failover on a single cluster node is also coordinated across the cluster so that all the nodes continue to share the same physical path.

Prior to release 4.1 of VxVM, the clustering and DMP features could not handle automatic failback in A/P arrays when a path was restored, and did not support failback for explicit failover mode arrays. Failback could only be implemented manually by running the `vxdctl enable` command on each cluster node after the path failure had been corrected. From release 4.1, failback is now an automatic cluster-wide operation that is coordinated by the master node. Automatic failback in explicit failover mode arrays is also handled by issuing the appropriate low-level command.

Note: Support for automatic failback of an A/P array requires that an appropriate ASL (and APM, if required) is available for the array, and has been installed on the system.

See "Discovering disks and dynamically adding disk arrays" on page 135.

For Active/Active type disk arrays, any disk can be simultaneously accessed through all available physical paths to it. In a clustered environment, the nodes do not all need to access a disk via the same physical path.

See "How to administer the Device Discovery Layer" on page 139.

See "Configuring array policy modules" on page 131.

About enabling or disabling controllers with shared disk groups

Prior to release 5.0, VxVM did not allow enabling or disabling of paths or controllers connected to a disk that is part of a shared Veritas Volume Manager disk group. From VxVM 5.0 onward, such operations are supported on shared DMP nodes in a cluster.

Multiple paths to disk arrays

Some disk arrays provide multiple ports to access their disk devices. These ports, coupled with the host bus adaptor (HBA) controller and any data bus or I/O processor local to the array, make up multiple hardware paths to access the disk devices. Such disk arrays are called multipathed disk arrays. This type of disk array can be connected to host systems in many different configurations, (such as multiple ports connected to different controllers on a single host, chaining of the ports through a single controller on a host, or ports connected to different hosts simultaneously).

See "How DMP works" on page 14.

Device discovery

Device discovery is the term used to describe the process of discovering the disks that are attached to a host. This feature is important for DMP because it needs to support a growing number of disk arrays from a number of vendors. In conjunction with the ability to discover the devices attached to a host, the Device Discovery service enables you to add support dynamically for new disk arrays. This operation, which uses a facility called the Device Discovery Layer (DDL), is achieved without the need for a reboot.

This means that you can dynamically add a new disk array to a host, and run a command which scans the operating system's device tree for all the attached disk devices, and reconfigures DMP with the new device database.

See "How to administer the Device Discovery Layer" on page 139.

Disk devices

The device name (sometimes referred to as devname or disk access name) defines the name of a disk device as it is known to the operating system.

Such devices are usually, but not always, located in the `/dev` directory. Devices that are specific to hardware from certain vendors may use their own path name conventions.

DMP uses the device name to create metadevices in the `/dev/vx/[r]dmp` directories. Dynamic Multi-Pathing (DMP) uses the metadevices (or DMP nodes) to represent disks that can be accessed by one or more physical paths, perhaps via different controllers. The number of access paths that are available depends on whether the disk is a single disk, or is part of a multiported disk array that is connected to a system.

You can use the `vxdisk` utility to display the paths that are subsumed by a DMP metadevice, and to display the status of each path (for example, whether it is enabled or disabled).

See "How DMP works" on page 14.

Device names may also be remapped as enclosure-based names.

See "Disk device naming in DMP" on page 24.

Disk device naming in DMP

Device names for disks are assigned according to the naming scheme which you specify to DMP. The format of the device name may vary for different categories of disks.

See "Disk categories" on page 136.

Device names can use one of the following naming schemes:

- Operating system-based naming
- Enclosure-based naming

Devices with device names longer than 31 characters always use enclosure-based names.

By default, DMP uses enclosure-based naming.

You can change the disk-naming scheme if required.

See "Changing the disk-naming scheme" on page 150.

Operating system-based naming

In the OS-based naming scheme, all disk devices are named using the `hdisk#` format, where # is a series number.

DMP assigns the name of the DMP meta-device (disk access name) from the multiple paths to the disk. DMP sorts the names by hdisk number, and selects the smallest number. For example, hdisk1 rather than hdisk2.This behavior make it easier to correlate devices with the underlying storage.

If a CVM cluster is symmetric, each node in the cluster accesses the same set of disks. This naming scheme makes the naming consistent across nodes in a symmetric cluster.

By default, OS-based names are not persistent, and are regenerated if the system configuration changes the device name as recognized by the operating system. If you do not want the OS-based names to change after reboot, set the persistence attribute for the naming scheme.

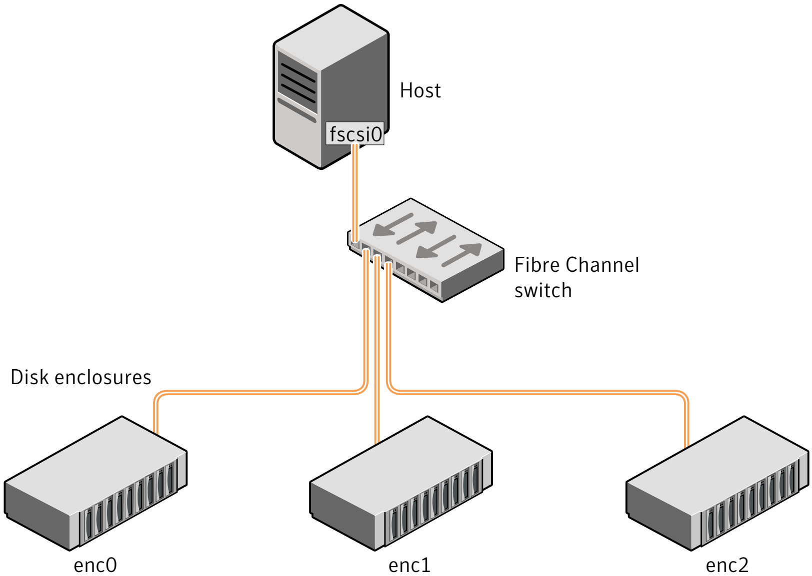
See “Changing the disk-naming scheme” on page 150.

About enclosure-based naming

In a Storage Area Network (SAN) that uses Fibre Channel switches, information about disk location provided by the operating system may not correctly indicate the physical location of the disks. Enclosure-based naming allows VxVM to access enclosures as separate physical entities. By configuring redundant copies of your data on separate enclosures, you can safeguard against failure of one or more enclosures.

Figure 1-3 shows a typical SAN environment where host controllers are connected to multiple enclosures through a Fibre Channel switch.

Figure 1-3 Example configuration for disk enclosures connected via a fibre channel switch



In such a configuration, enclosure-based naming can be used to refer to each disk within an enclosure. For example, the device names for the disks in enclosure `enc0` are named `enc0_0`, `enc0_1`, and so on. The main benefit of this scheme is that it allows you to quickly determine where a disk is physically located in a large SAN configuration.

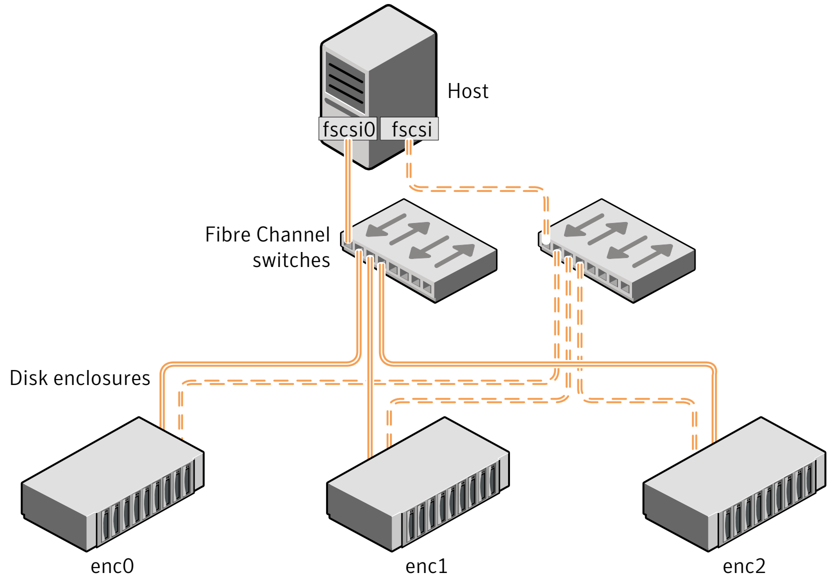
In most disk arrays, you can use hardware-based storage management to represent several physical disks as one LUN to the operating system. In such cases, VxVM also sees a single logical disk device rather than its component disks. For this reason, when reference is made to a disk within an enclosure, this disk may be either a physical disk or a LUN.

If required, you can replace the default name that VxVM assigns to an enclosure with one that is more meaningful to your configuration.

See "Renaming an enclosure" on page 124.

Figure 1-4 shows a High Availability (HA) configuration where redundant-loop access to storage is implemented by connecting independent controllers on the host to separate switches with independent paths to the enclosures.

Figure 1-4 Example HA configuration using multiple switches to provide redundant loop access



Such a configuration protects against the failure of one of the host controllers (`fscsi0` and `fscsi1`), or of the cable between the host and one of the switches. In this example, each disk is known by the same name to VxVM for all of the paths over which it can be accessed. For example, the disk device `enc0_0` represents a single disk for which two different paths are known to the operating system, such as `hdisk15` and `hdisk27`.

See "Disk device naming in DMP" on page 24.

See "Changing the disk-naming scheme" on page 150.

To take account of fault domains when configuring data redundancy, you can control how mirrored volumes are laid out across enclosures.

Enclosure-based naming

By default, DMP uses enclosure-based naming.

Enclosure-based naming operates as follows:

- All fabric or non-fabric disks in supported disk arrays are named using the *`enclosure_name_#`* format. For example, disks in the supported disk array, `enggdept` are named `enggdept_0`, `enggdept_1`, `enggdept_2` and so on.

  You can use the `vxdmpadm` command to administer enclosure names.

See "Renaming an enclosure" on page 124.

See the vxdmpadm(1M) manual page.

- Disks in the DISKS category (JBOD disks) are named using the Disk_# format.
- Devices in the OTHER_DISKS category are disks that are not multipathed by DMP. Devices in this category have names of the form hdisk#, which are the same as the device names generated by AIX.

By default, enclosure-based names are persistent, so they do not change after reboot.

If a CVM cluster is symmetric, each node in the cluster accesses the same set of disks. Enclosure-based names provide a consistent naming system so that the device names are the same on each node.

To display the native OS device names of a DMP disk (such as mydg01), use the following command:

```
# vxdisk path | grep diskname
```

See "Renaming an enclosure" on page 124.

See "Disk categories" on page 136.

Enclosure based naming with the Array Volume Identifier (AVID) attribute

By default, DMP assigns enclosure-based names to DMP meta-devices using an array-specific attribute called the Array Volume ID (AVID). The AVID provides a unique identifier for the LUN that is provided by the array. The ASL corresponding to the array provides the AVID property. Within an array enclosure, DMP uses the Array Volume Identifier (AVID) as an index in the DMP metanode name. The DMP metanode name is in the format *enclosureID_AVID*.

With the introduction of AVID to the EBN naming scheme, identifying storage devices becomes much easier. The array volume identifier (AVID) enables you to have consistent device naming across multiple nodes connected to the same storage. The disk access name never changes, because it is based on the name defined by the array itself.

Note: DMP does not support AVID with PowerPath names.

If DMP does not have access to a device's AVID, it retrieves another unique LUN identifier called the LUN serial number. DMP sorts the devices based on the LUN Serial Number (LSN), and then assigns the index number. All hosts see the same set of devices, so all hosts will have the same sorted list, leading to consistent device indices across the cluster. In this case, the DMP metanode name is in the format *enclosureID_index*.

DMP also supports a scalable framework, that allows you to fully customize the device names on a host by applying a device naming file that associates custom names with cabinet and LUN serial numbers.

If a CVM cluster is symmetric, each node in the cluster accesses the same set of disks. Enclosure-based names provide a consistent naming system so that the device names are the same on each node.

The DMP utilities such as `vxdisk list` display the DMP metanode name, which includes the AVID property. Use the AVID to correlate the DMP metanode name to the LUN displayed in the array management interface (GUI or CLI) .

For example, on an EMC CX array where the enclosure is emc_clariion0 and the array volume ID provided by the ASL is 91, the DMP metanode name is emc_clariion0_91. The following sample output shows the DMP metanode names:

```
$ vxdisk list
emc_clariion0_91  auto:cdsdisk  emc_clariion0_91   dg1  online shared
emc_clariion0_92  auto:cdsdisk  emc_clariion0_92   dg1  online shared
emc_clariion0_93  auto:cdsdisk  emc_clariion0_93   dg1  online shared
emc_clariion0_282 auto:cdsdisk  emc_clariion0_282  dg1  online shared
emc_clariion0_283 auto:cdsdisk  emc_clariion0_283  dg1  online shared
emc_clariion0_284 auto:cdsdisk  emc_clariion0_284  dg1  online shared
```

```
# vxddladm get namingscheme
NAMING_SCHEME       PERSISTENCE    LOWERCASE      USE_AVID
============================================================
Enclosure Based     Yes            Yes            Yes
```

Chapter 2

Setting up DMP to manage native devices

This chapter includes the following topics:

About setting up DMP to manage native devices

You can use DMP instead of third-party drivers for advanced storage management. This section describes how to set up DMP to manage native LVM devices and any logical volume that operates on those devices.

After you install DMP, set up DMP for use with LVM. To set up DMP for use with LVM, turn on the dmp_native_support tunable. When this tunable is turned on, DMP enables support for LVM on any device that does not have a VxVM label and is not in control of any third party multi-pathing (TPD) software. In addition, turning on the dmp_native_support tunable migrates any LVM volume groups that are not in use onto DMP devices.

The dmp_native_support tunable enables DMP support for LVM, as follows:

LVM volume groups	If the LVM volume groups are not in use, turning on native support migrates the devices to DMP devices. If the LVMvolume groups are in use, perform steps to turn off the devices and migrate the devices to DMP.
Veritas Volume Manager (VxVM) devices	Native support is not enabled for any device that has a VxVM label. To make the device available for LVM, remove the VxVM label. VxVM devices can coexist with native devices under DMP control.
Devices that are multipathed with Third-party drivers (TPD)	If a disk is already multipathed with a third-party driver (TPD), DMP does not manage the devices unless you remove TPD support. After you remove TPD support, turning on the `dmp_native_support` tunable migrates the devices. If you have LVM volume groups constructed over TPD devices, then you need to follow specific steps to migrate the LVM volume groups onto DMP devices.

To turn on the dmp_native_support tunable, use the following command:

```
# vxdmpadm settune dmp_native_support=on
```

The first time this operation is performed, the command reports if a volume group is in use, and does not migrate those devices. To migrate the volume group onto DMP, stop the volume group. Then execute the `vxdmpadm settune` command again to migrate the volume group onto DMP.

To verify the value of the dmp_native_support tunable, use the following command:

```
# vxdmpadm gettune dmp_native_support
         Tunable               Current Value  Default Value
----------------------------    ----------------    -----------------
dmp_native_support                on              off
```

Migrating LVM volume groups to DMP

You can use DMP instead of third-party drivers for advanced storage management. This section describes how to set up DMP to manage LVM volume groups and the file systems operating on them.

To set up DMP, migrate the devices from the existing third-party device drivers to DMP.

Table 2-1 shows the supported native solutions and migration paths.

Table 2-1 Supported migration paths

Operating system	Native solution	Migration procedure
AIX	EMC PowerPath	See "Migrating to DMP from EMC PowerPath" on page 33.
AIX	Hitachi Data Link Manager (HDLM)	See "Migrating to DMP from Hitachi Data Link Manager (HDLM)" on page 34.
AIX	IBM Multipath IO (MPIO)	See "Migrating to DMP from IBM Multipath IO (MPIO)" on page 35.
AIX	AIX IBM SDD (vpath)	See "Migrating to DMP from IBM SDD (vpath)" on page 36.

Migrating to DMP from EMC PowerPath

This procedure describes removing devices from EMC PowerPath control and enabling DMP on the devices.

Plan for system downtime for the following procedure.

The migration steps involve system downtime on a host due to the following:

- Need to stop applications
- Need to stop the VCS services if using VCS

To remove devices from EMC PowerPath control and enable DMP

1. Turn on the DMP support for the LVM volume group.

   ```
   # vxdmpadm settune dmp_native_support=on
   ```

2. Stop the applications that use the PowerPath meta-devices.

   In a VCS environment, stop the VCS service group of the application, which will stop the application.

3. Unmount any file systems that use the volume group on the PowerPath device.

4. Stop the LVM volume groups that use the PowerPath device.

   ```
   # varyoffvg vgroupname
   ```

5 If the root volume group (rootvg) is under PowerPath control, migrate the rootvg to DMP.

See "Migrating a SAN root disk from EMC PowerPath to DMP control" on page 77.

6 Remove the disk access names for the PowerPath devices from VxVM.

```
# vxdisk rm emcpowerXXXX
```

Where *emcpowerXXXX* is the name of the device.

7 Take the device out of PowerPath control:

```
# powermt unmanage dev=pp_device_name
# powermt unmanage class=array_class
```

8 Verify that the PowerPath device has been removed from PowerPath control.

```
# powermt display dev=all
```

9 Run a device scan to bring the devices under DMP control:

```
# vxdisk scandisks
```

10 Mount the file systems.

11 Restart the applications.

Migrating to DMP from Hitachi Data Link Manager (HDLM)

This procedure describes removing devices from HDLM control and enabling DMP on the devices.

Note: DMP cannot co-exist with HDLM; HDLM must be removed from the system.

Plan for system downtime for the following procedure.

The migration steps involve system downtime on a host due to the following:

- Need to stop applications
- Need to stop the VCS services if using VCS
- The procedure involves one or more host reboots

To remove devices from Hitachi Data Link Manager (HDLM) and enable DMP

1. Stop the applications using the HDLM meta-device
2. Unmount any file systems that use the volume group on the HDLM device.
3. Stop the LVM volume groups that use the HDLM device.

   ```
   # varyoffvg vgroupname
   ```

4. Uninstall the HDLM package.
5. Turn on the DMP support for the LVM volume group.

   ```
   # vxdmpadm settune dmp_native_support=on
   ```

   The above command also enables DMP root support.
6. Reboot the system.
7. After the reboot, DMP controls the devices. If there were any LVM volume groups on HDLM devices they are migrated onto DMP devices.
8. Mount the file systems.
9. Restart the applications.

Migrating to DMP from IBM Multipath IO (MPIO)

This procedure describes removing devices from MPIO control and enabling DMP on the devices.

Plan for system downtime for the following procedure.

The migration steps involve system downtime on a host due to the following:

- Need to stop applications
- Need to stop the VCS services if using VCS
- The procedure involves one or more host reboots

To take the devices out of MPIO control and enable DMP

1. Obtain the corresponding MPIO suppression ODM packages for the array from the array vendor.
2. Stop the applications that use the MPIO devices.
3. Unmount the file systems on the MPIO devices.
4. Vary off the LVM volume groups.

   ```
   # varyoffvg vgroupname
   ```

5 Install the ODM packages from the array vendor. Refer to the array vendor documentation for the installation procedure.

   Some array vendors do not distribute ODM Pre-defines for their arrays for AIX. In this case, you can use the devices as hdisk devices, as long as MPIO does not claim these LUNs.

6 Turn on the DMP support for the LVM volume groups. This command also enables DMP root support.

```
# vxdmpadm settune dmp_native_support=on
```

7 Reboot the system.

8 After the reboot, DMP controls the devices. Any LVM volume groups on MPIO devices are migrated onto DMP devices.

9 Mount the file systems.

10 Restart the applications.

Migrating to DMP from IBM SDD (vpath)

This procedure describes removing devices from SDD control and enabling DMP on the devices.

Plan for system downtime for the following procedure.

The migration steps involve system downtime on a host due to the following:

- Need to stop applications
- Need to stop the VCS services if using VCS
- The procedure involves one or more host reboots

To take the devices out of SDD control and enable DMP

1 Stop the applications that use SDD devices.

2 Unmount the file systems that use SDD devices.

3 Vary off the LVM volume groups.

```
# varyoff vgroupname
```

4 Stop the SDD server daemon

```
# stopsrc -s sddsrv
```

5 Verify that the SDD server has stopped.

```
# lssrc
```

6 Remove the SDD vpath devices:

```
# rmdev -dl dpo -R
vpath0 deleted
vpath1 deleted
...
```

7 Uninstall the SDD driver package devices.

```
# sdd.os-version.rte
```

Note: DO NOT uninstall the Host Attachments packages for the arrays that are controlled by SDD.

8 Turn on the DMP support for the LVM volume groups.

```
# vxdmpadm settune dmp_native_support=on
```

9 Reboot the system.

10 After the reboot, DMP controls the devices. Any LVM volume groups on SDD devices are migrated onto DMP devices.

11 Mount the file systems.

12 Restart the applications.

Using DMP devices with Oracle Automatic Storage Management (ASM)

This release of DMP supports using DMP devices with Oracle Automatic Storage (ASM). DMP supports the following operations:

- See "Enabling DMP devices for use with ASM" on page 38.
- See "Removing DMP devices from the listing of ASM disks" on page 38.
- See " Migrating ASM disk groups on operating system devices to DMP devices" on page 39.

Enabling DMP devices for use with ASM

Enable DMP support for ASM to make DMP devices visible to ASM as available disks.

To make DMP devices visible to ASM

1 From ASM, make sure ASM_DISKSTRING is set to the value `/dev/vx/rdmp/*`.

```
SQL> show parameter ASM_DISKSTRING;
 NAME                 TYPE        VALUE
 -------------------- ----------- ---------------
 asm_diskstring       string      /dev/vx/rdmp/*
```

2 As root user, enable DMP devices for use with ASM.

```
# vxdmpasm enable username groupname [devicename ...]
```

For example:

```
# vxdmpasm enable oracle dba eva4k6k0_1
```

3 From ASM, confirm that ASM can see these new devices.

```
SQL> select name,path,header_status from v$asm_disk;

  NAME PATH HEADER_STATU
  ---------------------------------------------------
  ... .......     ....
   /dev/vx/rdmp/eva4k6k0_1   CANDIDATE
  ... .......     ....
```

Removing DMP devices from the listing of ASM disks

To remove DMP devices from the listing of ASM disks, disable DMP support for ASM from the device. You cannot remove DMP support for ASM from a device that is in an ASM disk group.

To remove the DMP device from the listing of ASM disks

1. If the device is part of any ASM disk group, remove the device from the ASM disk group.
2. As root user, disable DMP devices for use with ASM.

   ```
   # vxdmpasm disable diskname
   ```

   For example:

   ```
   # vxdmpasm disable eva4k6k0_1
   ```

Migrating ASM disk groups on operating system devices to DMP devices

When an existing ASM disk group uses operating system native devices as disks, you can migrate these devices to Veritas Dynamic Multi-Pathing control. If the OS devices are controlled by other multi-pathing drivers, this operation requires system downtime to migrate the devices to DMP control.

After this procedure, the ASM disk group uses the migrated DMP devices as its disks.

"From ASM" indicates that you perform the step as the user running the ASM instance.

"As root user" indicates that you perform the step as the root user.

To migrate an ASM disk group from operating system devices to DMP devices

1. From ASM, identify the ASM disk group that you want to migrate, and identify the disks under its control.
2. From ASM, dismount the ASM disk group.
3. If the devices are controlled by other multi-pathing drivers such as MPIO or PowerPath, migrate the devices to DMP control. Perform these steps as root user.

   See "About setting up DMP to manage native devices" on page 31.

4 As root user, enable DMP support for the ASM disk group identified in step 1.

```
# vxdmpasm enable username
   groupname [devicename ...]
```

Where *username* represents the ASM user running the ASM instance, and *groupname* represents the UNIX groupname of the specified user-id. If you specify one or more *devicenames*, DMP support for ASM is enabled for those devices. If you do not specify a *devicename*, DMP support is enabled for all devices in the system that have an ASM signature.

5 From ASM, set ASM_DISKSTRING to the value `/dev/vx/rdmp/*`

6 From ASM, confirm that the devices are available to ASM.

7 From ASM, mount the ASM disk groups. The disk groups are mounted on DMP devices.

Example: To migrate an ASM disk group from operating system devices to DMP devices

1 From ASM, identify the ASM disk group that you want to migrate, and identify the disks under its control.

```
SQL> select name, state from v$asm_diskgroup;
NAME                           STATE
------------------------------ -----------
ASM_DG1                        MOUNTED

SQL> select name,path,header_status from v$asm_disk;
NAME         PATH               HEADER_STATU
-------------------------------------------
ASM_DG1_0000 /dev/rhdisk43      MEMBER
ASM_DG1_0001 /dev/rhdisk51      MEMBER
ASM_DG1_0002 /dev/rhdisk97      MEMBER
```

2 From ASM, dismount the ASM disk group.

```
SQL> alter diskgroup ASM_DG1 dismount;
Diskgroup altered.

SQL> select name , state from v$asm_diskgroup;
NAME                           STATE
------------------------------ -----------
ASM_DG1                        DISMOUNTED
```

3 If the devices are controlled by other multi-pathing drivers, migrate the devices to DMP control. Perform these steps as root user.

Note: This step requires planned downtime of the system.

See "Setting up DMP to manage native devices"

4 As root user, enable DMP support for the ASM disk group identified in step 1, in one of the following ways:

- To migrate selected ASM diskgroups, use the `vxdmpadm` command to determine the DMP nodes that correspond to the OS devices.

  ```
  # vxdmpadm getdmpnode nodename=hdisk4
  NAME        STATE   ENCLR-TYPE PATHS ENBL DSBL ENCLR-NAME
  =========================================================
  EVA4K6K0_0 ENABLED EVA4K6K    4     4    0    EVA4K6K0
  ```

  Use the device name in the command below:

  ```
  # vxdmpasm enable oracle dba eva4k6k0_0 \
       eva4k6k0_9 emc_clariion0_243
  ```

- If you do not specify a *devicename*, DMP support is enabled for all devices in the disk group that have an ASM signature. For example:

  ```
  # vxdmpasm enable oracle dba
  ```

5 From ASM, set ASM_DISKSTRING to the value `/dev/vx/rdmp/*`.

```
SQL> alter system set ASM_DISKSTRING='/dev/vx/rdmp/*';
System altered.
SQL> show parameter ASM_DISKSTRING;
NAME                           TYPE       VALUE
-------------------------- --------- -------------------
asm_diskstring                 string     /dev/vx/rdmp/*
```

6 From ASM, confirm that the devices are available to ASM.

```
SQL> select name,path,header_status from v$asm_disk where
header_status='MEMBER';
NAME      PATH                                HEADER_STATU
-----------------------------------------------------------
          /dev/vx/rdmp/emc_clariion0_243     MEMBER
          /dev/vx/rdmp/eva4k6k0_9            MEMBER
          /dev/vx/rdmp/eva4k6k0_1            MEMBER
```

7 From ASM, mount the ASM disk groups. The disk groups are mounted on DMP devices.

```
SQL> alter diskgroup ASM_DG1 mount;
Diskgroup altered.

SQL> select name, state from v$asm_diskgroup;
NAME                           STATE
------------------------------ -----------
ASM_DG1                        MOUNTED

SQL>  select name,path,header_status from v$asm_disk where
header_status='MEMBER';
NAME          PATH                               HEADER_STATU
------------------------------------------------------------
ASM_DG1_0002 /dev/vx/rdmp/emc_clariion0_243  MEMBER
ASM_DG1_0000 /dev/vx/rdmp/eva4k6k0_1         MEMBER
ASM_DG1_0001 /dev/vx/rdmp/eva4k6k0_9         MEMBER
```

Adding DMP devices to an existing LVM volume group or creating a new LVM volume group

When the dmp_native_support is ON, you can create a new LVM volume group on an available DMP device. You can also add an available DMP device to an existing LVM volume group. After the LVM volume groups are on DMP devices, you can use any of the LVM commands to manage the volume groups.

To create a new LVM volume group on a DMP device or add a DMP device to an existing LVM volume group

1 Choose disks that are available for use by LVM. The `vxdisk list` command displays disks that are not in use by VxVM with the TYPE auto:none and the STATUS Online invalid.

```
# vxdisk list


DEVICE            TYPE       DISK    GROUP    STATUS
. . .
emc_clariion0_84  auto:none  -       -        online invalid
emc_clariion0_85  auto:none  -       -        online invalid
```

2 Identify the ODM device name that corresponds to the device. The ODM device name is a truncated form of the DMP device name, since the ODM database requires a shorter name. The dmpname is an attribute of the ODM device name.

In this example, the DMP device name is emc_clariion0_84, and the ODM device name is emc_clari0_84. The enclosure index and the array volume ID (AVID) in the enclosure based name (EBN) are retained from the DMP device name.

You can use an ODM query such as the following to determine the ODM device name:

```
# odmget -q "attribute = dmpname AND value = emc_clariion0_84"
CuAt

CuAt:
        name = "emc_clari0_84"
        attribute = "dmpname"
        value = "emc_clariion0_84"
        type = "R"
        generic = "DU"
        rep = "s"
        nls_index = 2

# lspv
emc_clari0_84       none                      None
emc_clari0_85       none                      None

# lsdev -Cc disk
. . .
emc_clari0_84     Available          Veritas DMP Device
emc_clari0_85     Available          Veritas DMP Device

# lsattr -El emc_clari0_84
dmpname    emc_clariion0_84  DMP Device name            True
pvid       none              Physical volume identifier True
unique_id DGC%5FRAID%200%5FCK200080300687%5F600601601C101F0
0E5CF099D7209DE11 Unique device identifier   True
```

3 Create a new LVM volume group on a DMP device.

Use the ODM device name to specify the DMP device.

```
# mkvg -y newvg emc_clari0_84
0516-1254 mkvg: Changing the PVID in the ODM.
newvg

# lspv
emc_clari0_84       00c95c90837d5ff8      newvg active
emc_clari0_85       none                  None
```

4 Add a DMP device to an existing LVM volume group.

Use the ODM device name to specify the DMP device.

```
# extendvg -f newvg emc_clari0_85
0516-1254 mkvg: Changing the PVID in the ODM.

# lspv
emc_clari0_84       00c95c90837d5ff8      newvg active
emc_clari0_85       00c95c90837d612f      newvg active
```

5 Run the following command to trigger DMP discovery of the devices:

```
# vxdisk scandisks
```

6 After the discovery completes, the disks are shown as in use by LVM:

```
# vxdisk list

. . .
emc_clariion0_84 auto:LVM        -        -    LVM
emc_clariion0_85 auto:LVM        -        -    LVM
```

Displaying the native multi-pathing configuration

When DMP is enabled for native devices, the dmp_native_support attribute displays as ON. When the tunable is ON, all DMP disks are available for native volumes except:

- Devices that have a VxVM label
  If you initialize a disk for VxVM use, then the native multi-pathing feature is automatically disabled for the disk. When the VxVM label is removed, the native multi-pathing is enabled.

- Devices that are multi-pathed with Third-party drivers
  If a disk is already multi-pathed with a third-party driver (TPD), DMP does not manage the devices unless TPD support is removed.

To display whether DMP is enabled

1. Display the attribute `dmp_native_support`.

   ```
   # vxdmpadm gettune dmp_native_support
   ```

2. When the `dmp_native_support` tunable is ON, use the `vxdisk list` command to display available volumes. Volumes available to LVM display with the TYPE auto:none. Volumes that are already in use by LVM display with the TYPE auto:LVM.

Removing DMP support for native devices

The dmp_native_support tunable is persistent across reboots and fileset upgrades.

You can remove an individual device from control by LVM if you initialize it for VxVM, or if you set up TPD multi-pathing for that device.

To remove support for native devices from all DMP devices, turn off the dmp_native_support tunable.

This operation also disables DMP support for LVM rootvg, so it requires that you reboot the system. You can enable DMP support for the LVM rootvg separately, if required.

To turn off the dmp_native support tunable:

```
# vxdmpadm settune dmp_native_support=off
```

To view the value of the dmp_native_support tunable:

```
# vxdmpadm gettune dmp_native_support
Tunable                Current Value      Default Value
---------------------- ----------------   --------------
dmp_native_support           off               off
```

To retain DMP support for LVM rootvg after the dmp_native_support tunable is turned off, use the following command:

```
# vxdmpadm native enable vgname=rootvg
```

Chapter 3

Veritas Dynamic Multi-Pathing for the Virtual I/O Server

This chapter includes the following topics:

Virtual I/O server overview

Virtual I/O (VIO) server is a virtualization technology by IBM. A Virtual I/O server is a logical partition (LPAR) that runs a trimmed-down version of the AIX operating system. Virtual I/O servers have APV support, which allows sharing of physical I/O resources between virtual I/O clients.

See the PowerVM wiki for more in-depth information about VIO server and virtualization:

http://www.ibm.com/developerworks/wikis/display/virtualization/VIO

For more information, see the *PowerVM Virtualization on IBM System p redbook*:

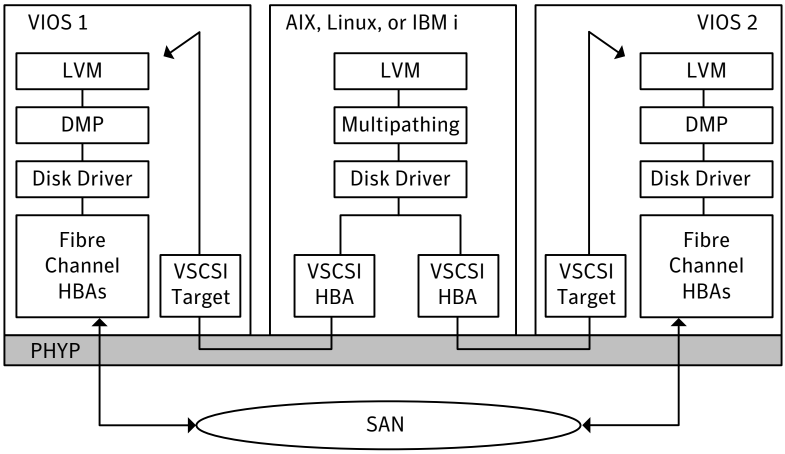
http://www.redbooks.ibm.com/redpieces/abstracts/sg247940.html

DMP support for Virtual I/O Server

DMP support in Virtual I/O Server requires a Veritas Dynamic Multi-Pathing (DMP) product license. Minimum VIOS oslevel required is 2.1.3 10-FP-23 or later.

Figure 3-1 illustrates DMP enablement in the Virtual I/O Server.

Figure 3-1 Veritas Dynamic Multi-Pathing in the Virtual I/O Server



DMP administration and management on Virtual I/O Server

DMP is fully functional in the Virtual I/O server. DMP administration and management commands (`vxdmpadm`, `vxddladm`, `vxdisk`, etc.) must be invoked from the non-restricted root shell.

```
$ oem_setup_env
```

Some example commands:

```
dmpvios1$ vxdmpadm getsubpaths dmpnodename=ibm_ds8x000_0337

NAME     STATE[A]  PATH-TYPE[M] CTLR-NAME ENCLR-TYPE ENCLR-NAME  ATTRS
==================================================================
hdisk21  ENABLED(A)  -          fscsi0    IBM_DS8x00  ibm_ds8x000  -
hdisk61  ENABLED(A)  -          fscsi0    IBM_DS8x00  ibm_ds8x000  -
```

```
hdisk80 ENABLED(A)  -         fscsi1   IBM_DS8x00  ibm_ds8x000  -
hdisk99 ENABLED(A)  -         fscsi1   IBM_DS8x00  ibm_ds8x000  -
```

```
dmpvios1$ vxdmpadm listenclosure all
```

```
ENCLR_NAME  ENCLR_TYPE ENCLR_SNO STATUS    ARRAY_TYPE LUN_COUNT
=================================================================
disk        Disk       DISKS     CONNECTED Disk       1
ibm_ds8x000 IBM_DS8x00 75MA641   CONNECTED A/A        6
```

For complete information about managing Dynamic Multi-Pathing, see the *Veritas Dynamic Multi-Pathing Administrator's Guide*.

Veritas Volume Manager (VxVM) administration and management

Veritas Volume Manager (VxVM) functionality is disabled in Virtual I/O Server. VxVM commands that manage volumes or disk groups are disabled in the VIO server.

In the VIOS, VxVM does not detect disk format information, so the disk status for VxVM disks is shown as unknown. For example:

```
dmpvios1$ vxdisk list
DEVICE           TYPE     DISK    GROUP   STATUS
disk_0           auto     -       -       unknown
ibm_ds8x000_02c1 auto     -       -       unknown
ibm_ds8x000_0288 auto     -       -       unknown
ibm_ds8x000_029a auto     -       -       unknown
ibm_ds8x000_0292 auto     -       -       unknown
ibm_ds8x000_0293 auto     -       -       unknown
ibm_ds8x000_0337 auto     -       -       unknown
```

In the VIOS, VxVM displays an error if you run a command that is disabled, as follows:

```
dmpvios1$ vxdisk -f init ibm_ds8x000_0288
VxVM vxdisk ERROR V-5-1-5433 Device ibm_ds8x000_0288: init failed:
Operation not allowed. VxVM is disabled.
```

```
dmpvios1$ vxdg import bootdg
VxVM vxdg ERROR V-5-1-10978 Disk group bootdg: import failed:
Operation not allowed. VxVM is disabled.
```

Configuring DMP on Virtual I/O Server

In this release, you can install DMP in the virtual I/O server (VIOS). This enables the VIO server to export dmpnodes to the VIO clients. The VIO clients access the dmpnodes in the same way as any other vSCSI devices. DMP handles the I/O to the disks backed by the dmpnodes.

Installing Veritas Dynamic Multi-Pathing (DMP) on Virtual I/O Server

Veritas Dynamic Multi-Pathing (DMP) can operate in the Virtual I/O server. Install DMP on the Virtual I/O server.

To install DMP on the Virtual I/O Server

1. Log into the VIO server partition.
2. Use the `oem_setup_env` command to access the non-restricted root shell.
3. Install Veritas Dynamic Multi-Pathing on the Virtual I/O Server.

   See the *Veritas Dynamic Multi-Pathing Installation Guide*.
4. Installing DMP on the VIO server enables the dmp_native_support tunable. Do not set the dmp_native_support tunable to off.

```
dmpvios1$ vxdmpadm gettune dmp_native_support
Tunable               Current Value  Default Value
------------------   --------------- -------------------
dmp_native_support       on              off
```

Migrating from other multi-pathing solutions to DMP on Virtual I/O Server

DMP supports migrating from AIX MPIO and EMC PowerPath multi-pathing solutions to DMP on Virtual I/O Server.

To migrate from other multi-pathing solutions to DMP on Virtual I/O Server

1. Before migrating, back up the Virtual I/O Servers to use for reverting the system in case of issues.
2. Shut down all VIO client partitions that are serviced by the VIOS.
3. Log into the VIO server partition. Use the following command to access the non-restricted root shell. All subsequent commands in this procedure must be invoked from the non-restricted shell.

```
$ oem_setup_env
```

4 For each Fibre Channel (FC) adapter on the system, verify that the following attributes have the recommended settings:

fc_err_recov	fast_fail
dyntrk	yes

If required, use the `chdev` command to change the attributes.

The following example shows how to change the attributes:

```
dmpvios1$ chdev -a fc_err_recov=fast_fail -a dyntrk=yes -l \
 fscsi0 -P
fscsi0 changed
```

The following example shows the new attribute values:

```
dmpvios1$ lsattr -El  fscsi0


attach       switch   How this adapter is CONNECTED  False
dyntrk       yes      Dynamic Tracking of FC Devices True
fc_err_recov fast_fail FC Fabric Event Error RECOVERY
Policy True
scsi_id      0xd0c00  Adapter SCSI ID                False
sw_fc_class  3        FC Class for Fabric            True
```

5 Use commands like `lsdev` and `lsmap` to view the configuration.

6 Unconfigure all VTD devices from all virtual adapters on the system:

```
dmpvios1$ rmdev -p vhost0
```

Repeat this step for all other virtual adapters.

7 Migrate from the third-party device driver to DMP.

Note that you do not need to do turn on the dmp_native_support again, because it is turned on for VIOS by default. You can use the `vxdmpadm gettune dmp_native_support` command to verify that the tunable parameter is turned on.

For the migration procedure, see the *Veritas Dynamic Multi-Pathing Administrator's Guide*.

8 Reboot the VIO Server partition.

9 Use the following command to verify that all Virtual SCSI mappings of TPD multi-pathing solution have been correctly migrated to DMP:

```
dmpvios1$ /usr/ios/cli/ioscli lsmap -all
```

10 Repeat step 1 through step 9 for all of the other VIO server partitions of the managed system.

11 After all of the VIO Server partitions are successfully migrated to DMP, start all of the VIO client partitions.

Example: migration from MPIO to DMP on Virtual I/O Server for a dual-VIOS configuration

This section shows an example of a migration from MPIO to DMP on the Virtual I/O Server, in a configuration with two VIO Servers.

Example configuration:

```
Managed System:  dmpviosp6
VIO server1: dmpvios1
VIO server2: dmpvios2
VIO clients: dmpvioc1
SAN LUNs: IBM DS8K array
Current multi-pathing solution on VIO server: IBM MPIO

ODM definition fileset required to disable MPIO support
for IBM DS8K array LUNs:
devices.fcp.disk.ibm.rte
```

To migrate dmpviosp6 from MPIO to DMP

1 Before migrating, back up the Virtual I/O Server to use for reverting the system in case of issues.

See the IBM website for information about backing up Virtual I/O Server.

2 Shut down all of the VIO clients that are serviced by the VIO Server.

```
dmpvioc1$ halt
```

3 Log into the VIO server partition.Use the following command to access the non-restricted root shell. All subsequent commands in this procedure must be invoked from the non-restricted shell.

```
$ oem_setup_env
```

4. Verify that the FC adapters have the recommended settings. If not, change the settings as required.

   For example, the following output shows the settings:

```
dmpvios1$ lsattr -El fscsi0
attach       switch  How this adapter is CONNECTED    False
dyntrk       yes     Dynamic Tracking of FC Devices   True
fc_err_recov fast_fail FC Fabric Event Error RECOVERY
Policy True
scsi_id      0xd0c00 Adapter SCSI ID                  False
sw_fc_class  3       FC Class for Fabric              True
```

5 The following command shows `lsmap` output before migrating MPIO VTD devices to DMP:

```
dmpvios1$ /usr/ios/cli/iosctl lsmap -all
SVSA            Physloc                        Client Partition ID
--------------- ---------------------------- ------------------
vhost0          U9117.MMA.0686502-V2-C11       0x00000004

VTD                 vtscsi0
Status              Available 8100000000000000
Backing device      hdisk21
LUN                 0x
Physloc             U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L4
003403700000000

VTD                 vtscsi1
Status              Available
LUN                 0x8200000000000000
Backing device      hdisk20
Physloc             U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L4
00240C100000000

VTD                 vtscsi2
Status              Available
LUN                 0x8300000000000000
Backing device      hdisk18
Physloc             U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L4
002409A00000000
```

The VIO Server has MPIO providing multi-pathing to these hdisks. The following commands show the configuration:

```
dmpvios1$ lsdev -Cc disk | egrep "hdisk21|hdisk20|hdisk18"

hdisk18 Available 02-08-02 MPIO Other FC SCSI Disk Drive
hdisk20 Available 02-08-02 MPIO Other FC SCSI Disk Drive
hdisk21 Available 02-08-02 MPIO Other FC SCSI Disk Drive
```

6 Unconfigure all VTD devices from all virtual adapters on the system:

```
dmpvios1 $ rmdev -p vhost0
vtscsi0 Defined
vtscsi1 Defined
vtscsi2 Defined
```

Repeat this step for all other virtual adapters.

7 Migrate the devices from MPIO to DMP.

Unmount the file system and varyoff volume groups residing on the MPIO devices.

Display the volume groups (vgs) in the configuration:

```
dmpvios1$ lsvg
rootvg
brunovg
```

```
dmpvios1 lsvg -p brunovg
```

```
brunovg:
PV_NAME PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
hdisk19 active   511       501      103..92..102..102..102
hdisk22 active   511       501      103..92..102..102..102
```

Use the `varyoffvg` command on all affected vgs:

```
dmpvios1$ varyoffvg brunovg
```

Install the IBMDS8K ODM definition fileset to remove IBM MPIO support for IBM DS8K array LUNs.

```
dmpvios1$ installp -aXd . devices.fcp.disk.ibm.rte
```

```
+-------------------------------------------------------+
              Pre-installation Verification...
+-------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...
Installation Summary
--------------------
Name                       Level    Part  Event  Result
------------------------------------------------------
devices.fcp.disk.ibm.rte  1.0.0.2 USR   APPLY  SUCCESS
devices.fcp.disk.ibm.rte  1.0.0.2 ROOT  APPLY  SUCCESS
```

8 Reboot VIO server1

```
dmpvios1$ reboot
```

9 After the VIO server1 reboots, verify that all of the existing volume groups on the VIO server1 and MPIO VTDs on the VIO server1 are successfully migrated to DMP.

```
dmpvios1 lsvg -p brunovg

brunovg:
PV_NAME          PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
ibm_ds8000_0292 active   511       501    103..92..102..102..102
ibm_ds8000_0293 active   511       501    103..92..102..102..102
```

Verify the vSCSI mappings of IBM DS8K LUNs on the migrated volume groups:

```
dmpvios1 lsmap -all

SVSA            Physloc                       Client Partition ID
--------------- ----------------------------- ------------------
vhost0          U9117.MMA.0686502-V2-C11      0x00000000
VTD                   vtscsi0
Status                Available
LUN                   0x8100000000000000
Backing device        ibm_ds8000_0337
Physloc

VTD                   vtscsi1
Status                Available
LUN                   0x8200000000000000
Backing device        ibm_ds8000_02c1
Physloc

VTD                   vtscsi2
Status                Available
LUN                   0x8300000000000000
Backing device        ibm_ds8000_029a
Physloc
```

10 Repeat step 1 through step 9 for VIO server2.

11 Start all of the VIO clients using HMC.

Example: migration from PowerPath to DMP on Virtual I/O Server for a dual-VIOS configuration

This section shows an example of a migration from PowerPath to DMP on the Virtual I/O Server, in a configuration with two VIO Servers.

Example configuration:

Managed System: dmpviosp6

VIO server1: dmpvios1

VIO server2: dmpvios2

VIO clients: dmpvioc1

SAN LUNs: EMC Clariion array

Current multi-pathing solution on VIO server: EMC PowerPath

To migrate dmpviosp6 from PowerPath to DMP

1. Before migrating, back up the Virtual I/O Server to use for reverting the system in case of issues.

   See the IBM website for information about backing up Virtual I/O Server.

2. Shut down all of the VIO clients that are serviced by the VIO Server.

   ```
   dmpvioc1$ halt
   ```

3. Log into the VIO server partition.Use the following command to access the non-restricted root shell. All subsequent commands in this procedure must be invoked from the non-restricted shell.

   ```
   $ oem_setup_env
   ```

4. Verify that the FC adapters have the recommended settings. If not, change the settings as required.

   For example, the following output shows the settings:

   ```
   dmpvios1$ lsattr -El fscsi0
   attach       switch    How this adapter is CONNECTED  False
   dyntrk       yes       Dynamic Tracking of FC Devices True
   fc_err_recov fast_fail FC Fabric Event Error RECOVERY Policy
    True
   scsi_id      0xd0c00   Adapter SCSI ID                False
   sw_fc_class  3         FC Class for Fabric            True
   ```

5 The following command shows `lsmap` output before migrating PowerPath VTD devices to DMP:

```
dmpvios1$ /usr/ios/cli/iosctl lsmap -all


SVSA            Physloc                      Client Partition ID
--------------- ---------------------------- --------------------
vhost0          U9117.MMA.0686502-V2-C11     0x00000004

VTD                P0
Status             Available
LUN                0x8100000000000000
Backing device     hdiskpower0
Physloc            U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L4
0034037
00000000

VTD               P1
Status            Available
LUN               0x8200000000000000
Backing device    hdiskpower1
Physloc           U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L40
0240C10
0000000

VTD               P2
Status            Available
LUN               0x8300000000000000
Backing device    hdiskpower2
Physloc           U789D.001.DQD04AF-P1-C5-T1-W500507630813861A-L40
02409A00000000
```

6 Unconfigure all VTD devices from all virtual adapters on the system:

```
dmpvios1 $ rmdev -p vhost0
P0 Defined
P1 Defined
P2 Defined
```

Repeat this step for all other virtual adapters.

7. Migrate the devices from PowerPath to DMP.

   Unmount the file system and varyoff volume groups residing on the PowerPath devices.

   Display the volume groups (vgs) in the configuration:

   ```
   dmpvios1$ lsvg
   rootvg
   brunovg
   ```

   ```
   dmpvios1 lsvg -p brunovg
   ```

   ```
   brunovg:
   PV_NAME     PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
   hdiskpower3 active   511       501    103..92..102..102..102
   ```

   Use the varyoffvg command on all affected vgs:

   ```
   dmpvios1$ varyoffvg brunovg
   ```

   Unmanage the EMC Clariion array from PowerPath control

   ```
   # powermt unmanage class=clariion
   hdiskpower0 deleted
   hdiskpower1 deleted
   hdiskpower2 deleted
   hdiskpower3 deleted
   ```

8. Reboot VIO server1

   ```
   dmpvios1$ reboot
   ```

9 After the VIO server1 reboots, verify that all of the existing volume groups on the VIO server1 and MPIO VTDs on the VIO server1 are successfully migrated to DMP.

```
dmpvios1 lsvg -p brunovg

brunovg:
PV_NAME       PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
emc_clari0_138 active  511       501    103..92..102..102..102
```

Verify the mappings of the LUNs on the migrated volume groups:

```
dmpvios1 lsmap -all

SVSA            Physloc                     Client Partition ID
--------------- --------------------------- ------------------
vhost0          U9117.MMA.0686502-V2-C11    0x00000000
VTD                   P0
Status                Available
LUN                   0x8100000000000000
Backing device        emc_clari0_130
Physloc

VTD                   P1
Status                Available
LUN                   0x8200000000000000
Backing device        emc_clari0_136
Physloc

VTD                   P2
Status                Available
LUN                   0x8300000000000000
Backing device        emc_clari0_137
Physloc
```

10 Repeat step 1 to step 9 for VIO server2.

11 Start all of the VIO clients.

Configuring DMP pseudo devices as virtual SCSI devices

DMP in the VIO server supports the following methods to export a device to the VIO client:

- DMP node method
  See "Exporting DMP devices as Virtual SCSI disks " on page 62.
- Logical partition-based method
  See "Exporting a Logical Volume as a Virtual SCSI disk" on page 65.
- File-based method
  See "Exporting a file as a virtual SCSI disk" on page 67.

Exporting DMP devices as Virtual SCSI disks

DMP supports disks backed by DMP as Virtual SCSI disks. Export the DMP device as a vSCSI disk to the VIO client.

To export a DMP device as a vSCSI disk

1. Log into the VIO server partition.Use the following command to access the non-restricted root shell. All subsequent commands in this procedure must be invoked from the non-restricted shell.

   ```
   $ oem_setup_env
   ```

2. Use the `oem_setup_env` command to access the non-restricted root shell.
3. The following command displays the DMP devices on the VIO server:

   ```
   dmpvios1$ lsdev -t dmpdisk

   ibm_ds8000_0287 Available  Veritas DMP Device
   ibm_ds8000_0288 Available  Veritas DMP Device
   ibm_ds8000_0292 Available  Veritas DMP Device
   ibm_ds8000_0293 Available  Veritas DMP Device
   ibm_ds8000_029a Available  Veritas DMP Device
   ibm_ds8000_02c1 Available  Veritas DMP Device
   ibm_ds8000_0337 Available  Veritas DMP Device
   ```

4. Assign the DMP device as a backing device. Exit from the non-restricted shell to run this command from the VIOS default shell.

   ```
   dmpvios1$ exit

   $ mkvdev -vdev ibm_ds8000_0288 -vadapter vhost0
   vtscsi3 Available
   ```

5 Use the following command to display the configuration.

```
$ lsmap -all

SVSA            Physloc                        Client Partition ID
--------------- ------------------------------ ------------------
vhost0          U9117.MMA.0686502-V2-C11       0x00000000
VTD                   vtscsi0
Status                Available
LUN                   0x8100000000000000
Backing device        ibm_ds8000_0337
Physloc

VTD                   vtscsi1
Status                Available
LUN                   0x8200000000000000
Backing device        ibm_ds8000_02c1
Physloc

VTD                   vtscsi2
Status                Available
LUN                   0x8300000000000000
Backing device        ibm_ds8000_029a
Physloc V

TD                    vtscsi3
Status                Available
LUN                   0x8400000000000000
Backing device        ibm_ds8000_0288
Physloc
```

6 For a dual-VIOS configuration, export the DMP device corresponding to the same SAN LUN on the second VIO Server in the configuration. To export the DMP device on the second VIO server, identify the DMP device corresponding to the SAN LUN as on the VIO Server1.

- If the array supports the AVID attribute, the DMP device name is the same as the DMP device name on the VIO Server1.
- Otherwise, use the UDID value of the DMP device on the VIO Server1 to correlate the DMP device name with same UDID on the VIO Server2.
  On VIO Server1:

  ```
  $ oem_setup_env
  ```

```
dmpvios1$ lsattr -El ibm_ds8000_0288

attribute value            description        user_settable
dmpname   ibm_ds8x000_0288 DMP Device name   True
pvid      none             Physical volume identifier True
unique_id IBM%5F2107%5F75MA641%5F6005076308FFC61A000000000
0000288
Unique device identifier   True
```

On VIO Server2:

```
$ oem_setup_env

dmpvios2$ odmget -q "attribute = unique_id and
value = 'IBM%5F2107%5F75MA641%5F6005076308FFC61A000000000
0000288'" CuAt

CuAt:

   name = "ibm_ds8000_0288"
   attribute = "unique_id"
   value = "IBM%5F2107%5F75MA641%5F6005076308FFC61A00
   00000000000288"
   type = "R"
   generic = "DU"
   rep = "s"
   nls_index = 4
```

7 Use the DMP device name identified in step 6 to assign the DMP device as a backing device. Exit from the non-restricted shell to run this command from the VIOS default shell.

```
dmpvios1$ exit

$ mkvdev -vdev ibm_ds8000_0288 -vadapter vhost0
vtscsi3 Available
```

8 Use the following command to display the configuration.

```
$ lsmap -all

SVSA            Physloc                    Client Partition ID
-------------   -------------------------  -------------------
vhost0          U9117.MMA.0686502-V2-C11    0x00000000
VTD                   vtscsi0
Status                Available
LUN                   0x8100000000000000
Backing device        ibm_ds8000_0337
Physloc

VTD                   vtscsi1
Status                Available
LUN                   0x8200000000000000
Backing device        ibm_ds8000_02c1
Physloc

VTD                   vtscsi2
Status                Available
LUN                   0x8300000000000000
Backing device        ibm_ds8000_029a
Physloc V

TD                    vtscsi3
Status                Available
LUN                   0x8400000000000000
Backing device        ibm_ds8000_0288
Physloc
```

Exporting a Logical Volume as a Virtual SCSI disk

DMP supports vSCSI disks backed by a Logical Volume. Export the Logical Volume as a vSCSI disk to the VIO client.

To export a Logical Volume as a vSCSI disk

1. Create the volume group.

```
$ mkvg -vg brunovg ibm_ds8000_0292 ibm_ds8000_0293
brunovg
```

   The following command displays the new volume group:

```
$ lsvg -pv brunovg
brunovg:
PV_NAME          PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
ibm_ds8000_0292 active   494       494      99..99..98..99..99
ibm_ds8000_0293 active   494       494      99..99..98..99..99
```

2. Make a logical volume in the volume group.

```
$ mklv -lv brunovg_lv1 brunovg 1G
brunovg_lv1
```

   The following command displays the new logical volume:

```
$ lsvg -lv brunovg
brunovg:
LV NAME        TYPE   LPs   PPs   PVs  LV STATE      MOUNT POINT
brunovg_lv1    jfs    256   256   1    closed/syncd  N/A
```

3. Assign the logical volume as a backing device.

```
$ mkvdev -vdev brunovg_lv1 -vadapter  vhost0
vtscsi4 Available
```

4 Use the following command to display the configuration.

```
$ lsmap -all


SVSA           Physloc                   Client Partition ID
-------------- ------------------------- ------------------
vhost0         U9117.MMA.0686502-V2-C11  0x00000000
VTD                  vtscsi0
Status               Available
LUN                  0x8100000000000000
Backing device       ibm_ds8000_0337
Physloc

VTD                  vtscsi1
Status               Available
LUN                  0x8200000000000000
Backing device       ibm_ds8000_02c1
Physloc

VTD                  vtscsi2
Status               Available
LUN                  0x8300000000000000
Backing device       ibm_ds8000_029a
Physloc

VTD                  vtscsi3
Status               Available
LUN                  0x8400000000000000
Backing device       ibm_ds8000_0288
Physloc

VTD                  vtscsi4
Status               Available
LUN                  0x8500000000000000
Backing device       brunovg_lv1
Physloc
```

Exporting a file as a virtual SCSI disk

DMP supports vSCSI disks backed by a file. Export the file as a vSCSI disk to the VIO client.

To export a file as a vSCSI disk

1. Create the storage pool.

   ```
   $ mksp brunospool ibm_ds8000_0296
   brunospool
   0516-1254 mkvg: Changing the PVID in the ODM.
   ```

2. Create a file system on the pool.

   ```
   $ mksp -fb bruno_fb -sp brunospool -size 500M
   bruno_fb
   File system created successfully.
   507684 kilobytes total disk space.
   New File System size is 1024000
   ```

3. Mount the file system.

   ```
   $ mount


   node       mounted   mounted over   vfs    date     options
   ---------- ----------------------   ----- --------------------
   /dev/hd4       /        jfs2      Jul 02 14:47 rw,log=/dev/hd8
   /dev/hd2      /usr      jfs2      Jul 02 14:47 rw,log=/dev/hd8
   /dev/hd9var   /var      jfs2      Jul 02 14:47 rw,log=/dev/hd8
   /dev/hd3      /tmp      jfs2      Jul 02 14:47 rw,log=/dev/hd8
   /dev/hd1      /home     jfs2      Jul 02 14:48 rw,log=/dev/hd8
   /dev/hd11admin /admin   jfs2      Jul 02 14:48 rw,log=/dev/hd8
   /proc         /proc     procfs    Jul 02 14:48 rw
   /dev/hd10opt  /opt      jfs2      Jul 02 14:48 rw,log=/dev/hd8
   /dev/livedump /var/adm/ras/livedump jfs2 Jul 02 14:48 rw,log=
   /dev/hd8
   /dev/bruno_fb /var/vio/storagepools/bruno_fb jfs2 Jul 02 15:38
   rw,log=INLINE
   ```

4. Create a file in the storage pool.

   ```
   $ mkbdsp -bd bruno_fbdev -sp bruno_fb 200M
   Creating file "bruno_fbdev" in storage pool "bruno_fb".
   bruno_fbdev
   ```

5 Assign the file as a backing device.

```
$ mkbdsp -sp bruno_fb -bd bruno_fbdev -vadapter vhost0
Assigning file "bruno_fbdev" as a backing device.
vtscsi5 Available
bruno_fbdev
```

6 Use the following command to display the configuration.

```
$ lsmap -all

SVSA            Physloc                      Client Partition ID
--------------- ---------------------------- -------------------
vhost0          U9117.MMA.0686502-V2-C11     0x00000000
...
...
VTD                   vtscsi5
Status                Available
LUN                   0x8600000000000000
Backing device        /var/vio/storagepools/bruno_fb/bruno_fbdev
Physloc
```

Extended attributes in VIO client for a Virtual SCSI disk

Using DMP in the Virtual I/O server enables the DMP in the VIO Client to receive the extended attributes for the LUN. This enables the client LPAR to view back-end LUN attributes such as thin, SSD, and RAID levels associated with the vSCSI devices.

For more information about extended attributes and the prerequisites for supporting them, see the following tech note:

http://seer.entsupport.symantec.com/docs/337516.htm

Configuration prerequisites for providing extended attributes on VIO client for Virtual SCSI disk

DMP in VIO client will provide extended attributes information of backend SAN LUN. The following conditions are prerequisites for using extended attributes on the VIO client:

- VIO client has vSCSI disks backed by SAN LUNs.
- In the VIO Server partition, DMP is controlling those SAN LUNs.
- On VIO client, DMP is controlling the vSCSI disks.

Displaying extended attributes of Virtual SCSI disks

When a VIO client accesses a virtual SCSI disk that is backed by a DMP device on the Virtual I/O Server, the VIO client can access the extended attributes associated with the virtual SCSI disk.

The following commands can access and display extended attributes information associated with the vSCSI disk backed by DMP device on Virtual I/O Server.

- `vxdisk -e list`
- `vxdmpadm list dmpnodename=<daname>`
- `vxdmpadm -v getdmpnode dmpnodename=<daname>`
- `vxdisk -p list <daname>`

For example, use the following command on the VIO client dmpvioc1:

```
# vxdisk -e list
```

```
DEVICE           TYPE          DISK   GROUP  STATUS        OS_NATIVE_NAME  ATTR
ibm_ds8x000_114f auto:LVM      -      -      LVM           hdisk83         std
3pardata0_3968   auto:aixdisk  -      -      online thin   hdisk84         tp
```

```
# vxdmpadm list dmpnode dmpnodename=3pardata0_3968
```

```
dmpdev          = 3pardata0_3968
state           = enabled
enclosure       = 3pardata0
cab-sno         = 744
asl             = libvxvscsi.so
vid             = AIX
pid             = VDASD
array-name      = 3PARDATA
array-type      = VSCSI
iopolicy        = Single-Active
avid            = 3968
lun-sno         = 3PARdata%5FVV%5F02E8%5F2AC00F8002E8
udid            = AIX%5FVDASD%5F%5F3PARdata%255FVV%255F02E8%255F2AC00F8002E8
dev-attr        = tp
###path         = name state type transport ctlr hwpath aportID aportWWN attr
path            = hdisk84 enabled(a) - SCSI vscsi1 vscsi1 3 - -
```

Chapter 4

Administering DMP

This chapter includes the following topics:

About enabling and disabling I/O for controllers and storage processors

DMP allows you to turn off I/O for a controller or the array port of a storage processor so that you can perform administrative operations. This feature can be used for maintenance of HBA controllers on the host, or array ports that are attached to disk arrays supported by DMP. I/O operations to the controller or array port can be turned back on after the maintenance task is completed. You can accomplish these operations using the `vxdmpadm` command.

For Active/Active type disk arrays, after disabling the I/O through an HBA controller or array port, the I/O continues on the remaining paths. For Active/Passive type disk arrays, if disabling I/O through an HBA controller or array port resulted in all primary paths being disabled, DMP will failover to active secondary paths and I/O will continue on them.

After the operation is over, you can use `vxdmpadm` to re-enable the paths through the controllers.

See "Disabling I/O for paths, controllers or array ports" on page 122.

See "Enabling I/O for paths, controllers or array ports" on page 123.

You can also perform certain reconfiguration operations dynamically online.

See "About online dynamic reconfiguration" on page 155.

About displaying DMP database information

You can use the `vxdmpadm` command to list DMP database information and perform other administrative tasks. This command allows you to list all controllers that are connected to disks, and other related information that is stored in the DMP database. You can use this information to locate system hardware, and to help you decide which controllers need to be enabled or disabled.

The `vxdmpadm` command also provides useful information such as disk array serial numbers, which DMP devices (disks) are connected to the disk array, and which paths are connected to a particular controller, enclosure or array port.

See "Administering DMP using vxdmpadm" on page 94.

Displaying the paths to a disk

The `vxdisk` command is used to display the multi-pathing information for a particular metadevice. The metadevice is a device representation of a particular physical disk having multiple physical paths from one of the system's HBA controllers. In DMP, all the physical disks in the system are represented as metadevices with one or more physical paths.

To display the multi-pathing information on a system

- Use the `vxdisk path` command to display the relationships between the device paths, disk access names, disk media names and disk groups on a system as shown here:

```
# vxdisk path

SUBPATH      DANAME      DMNAME      GROUP    STATE
hdisk1       hdisk1      mydg01      mydg     ENABLED
hdisk9       hdisk9      mydg01      mydg     ENABLED
hdisk2       hdisk2      mydg02      mydg     ENABLED
hdisk10      hdisk10     mydg02      mydg     ENABLED
   .
   .
   .
```

This shows that two paths exist to each of the two disks, `mydg01` and `mydg02`, and also indicates that each disk is in the `ENABLED` state.

To view multi-pathing information for a particular metadevice

1 Use the following command:

```
# vxdisk list devicename
```

For example, to view multi-pathing information for `hdisk18`, use the following command:

```
# vxdisk list hdisk18
```

The output from the `vxdisk list` command displays the multi-pathing information, as shown in the following example:

```
Device:    hdisk18
devicetag: hdisk18
type:      simple
hostid:    system01
.
.
.
Multipathing information:
numpaths:   2
hdisk18 state=enabled  type=secondary
hdisk26 state=disabled type=primary
```

The `numpaths` line shows that there are 2 paths to the device. The next two lines in the "Multipathing information" section show that one path is active (`state=enabled`) and that the other path has failed (`state=disabled`).

The `type` field is shown for disks on Active/Passive type disk arrays such as the EMC CLARiiON, Hitachi HDS 9200 and 9500, Sun StorEdge 6xxx, and Sun StorEdge T3 array. This field indicates the primary and secondary paths to the disk.

The `type` field is not displayed for disks on Active/Active type disk arrays such as the EMC Symmetrix, Hitachi HDS 99xx and Sun StorEdge 99xx Series, and IBM ESS Series. Such arrays have no concept of primary and secondary paths.

2 Alternately, you can use the following command to view multi-pathing information:

```
# vxdmpadm getsubpaths dmpnodename=devicename
```

For example, to view multi-pathing information for `emc_clariion0_17`, use the following command:

```
# vxdmpadm getsubpaths dmpnodename=emc_clariion0_17
```

Typical output from the `vxdmpadm getsubpaths` command is as follows:

```
NAME     STATE[A]   PATH-TYPE[M] CTLR-NAME ENCLR-TYPE   ENCLR-NAME    ATTRS
===========================================================================
hdisk107 ENABLED(A) PRIMARY      fscsi1    EMC_CLARiiON emc_clariion0    -
hdisk17  ENABLED    SECONDARY    fscsi0    EMC_CLARiiON emc_clariion0    -
hdisk2   ENABLED    SECONDARY    fscsi0    EMC_CLARiiON emc_clariion0    -
hdisk32  ENABLED(A) PRIMARY      fscsi0    EMC_CLARiiON emc_clariion0    -
```

Setting customized names for DMP nodes

The DMP node name is the meta device name which represents the multiple paths to a disk. The DMP node name is generated from the device name according to the DMP naming scheme.

See "Disk device naming in DMP" on page 24.

You can specify a customized name for a DMP node. User-specified names are persistent even if names persistence is turned off.

You cannot assign a customized name that is already in use by a device. However, if you assign names that follow the same naming conventions as the names that the DDL generates, a name collision can potentially occur when a device is added. If the user-defined name for a DMP device is the same as the DDL-generated name for another DMP device, the `vxdisk list` command output displays one of the devices as 'error'.

To specify a custom name for a DMP node

◆ Use the following command:

```
# vxdmpadm setattr dmpnode dmpnodename name=name
```

You can also assign names from an input file. This enables you to customize the DMP nodes on the system with meaningful names.

To assign DMP nodes from a file

1. Use the script `vxgetdmpnames` to get a sample file populated from the devices in your configuration. The sample file shows the format required and serves as a template to specify your customized names.
2. To assign the names, use the following command:

   ```
   # vxddladm assign names file=pathname
   ```

To clear custom names

- To clear the names, and use the default OSN or EBN names, use the following command:

  ```
  # vxddladm -c assign names
  ```

Configuring DMP for SAN booting

The procedures in this section describe configuring DMP for SAN booting. The SAN boot disk must be an Active/Active (A/A), A/A-A, or ALUA type array.

Migrating an internal root disk to a SAN root disk under DMP control

If the system has been booted from an internal disk (such as hdisk0), you can configure an alternate root disk on the attached SAN storage before you put it under DMP control.

To migrate a root disk from LVM to DMP control

1. Use the `chdev` command to clear the PVID of the disk in the SAN storage that is to be the SAN root disk:

   ```
   # chdev -l hdisk373 -a pv=clear
   ```

   Repeat this command for all paths of the SAN root disk.
2. If any path to the target disk has the `reserve_policy` attribute, set the attribute to `no_reserve` for all the paths. This settings releases SCSI-2 reservation from the paths on reboot.

   ```
   # lsattr -E1 hdisk373 | grep res
   reserve_policy single_path
   Reserve Policy True
   ```

   ```
   # chdev -l hdisk373 -a reserve_policy=no_reserve -P
   hdisk373 changed
   ```

3. Use the `alt_disk_install` command to create the SAN root disk.

   ```
   # alt_disk_install -C hdisk373
   ```

4. Reboot the system from the SAN root disk.

5. Configure DMP.

   See "Configuring DMP support for booting over a SAN" on page 78.

   See "Example of migrating an internal root disk to a SAN root disk under DMP control" on page 80.

Migrating a SAN root disk to DMP control

If the system has been booted from a SAN disk under MPIO control, MPIO must be disabled before DMP control can be enabled.

To migrate a SAN root disk from MPIO to DMP control

1. Disable MPIO by installing a device-specific ODM definition fileset as described in the following TechNote:

   http://library.veritas.com/docs/263558

2. Reboot the system. The system is booted without any multipathing support.

3. Configure DMP.

   See "Configuring DMP support for booting over a SAN" on page 78.

Migrating a SAN root disk from EMC PowerPath to DMP control

If the system has a root volume group (rootvg) under EMC PowerPath control, use this procedure to migrate the rootvg to DMP control.

To migrate a SAN root disk from EMC PowerPath to DMP control

1. Remove the PowerPath device corresponding to the root disk (rootvg) from VxVM control:

   ```
   # vxdisk rm hdiskpowerX
   ```

2. Issue the following command so that PowerPath returns the pvid to the hdisk device. Otherwise bosboot does not succeed.

   ```
   # pprootdev fix
   ```

3 Remove the device from PowerPath so that PowerPath releases control of the boot device on the next reboot.

```
# powermt unmanage dev=hdiskpowerX
```

4 Enable DMP root support.

See "Configuring DMP support for booting over a SAN" on page 78.

5 Reboot the system. The system is booted with the rootvg under DMP control.

Configuring DMP support for booting over a SAN

Configuring DMP to work with an LVM root disk over a SAN requires that the system is correctly configured to use the boot device over all possible paths.

To configure DMP support for booting over a SAN

1. The PVID and volume group entries in the second and third columns should be identical for all the paths. Use the `lspv` command for the root volume group to verify that the PVID and volume group entries are set correctly.

   In this example, the LVM root disk is multipathed with 4 paths. The output from the `lspv` command for the root volume group (rootvg) is as follows:

   ```
   # lspv | grep rootvg
   hdisk374 00cbf5ce56def54d rootvg active
   hdisk375 00cbf5ce56def54d rootvg
   hdisk376 00cbf5ce56def54d rootvg
   hdisk377 00cbf5ce56def54d rootvg
   ```

2. If the PVID and volume group entries are not set correctly on any of the paths, use the `chdev` command to set the correct value.

   For example, the following output shows that the `hdisk377` path is not set correctly:

   ```
   # lspv | grep rootvg
   hdisk374 00cbf5ce56def54d rootvg active
   hdisk375 00cbf5ce56def54d rootvg
   hdisk376 00cbf5ce56def54d rootvg
   hdisk377 none             None
   ```

   To correct the setting for the path, use the following command:

   ```
   # chdev -l hdisk377 -a pv=yes
   hdisk377 changed
   ```

   The output of the `lspv` command now shows the correct values:

   ```
   # lspv | grep rootvg
   hdisk374 00cbf5ce56def54d rootvg active
   hdisk375 00cbf5ce56def54d rootvg
   hdisk376 00cbf5ce56def54d rootvg
   hdisk377 00cbf5ce56def54d rootvg
   ```

3. Check that the output from the `bootlist` command is correct:

```
# bootlist -m normal -o
hdisk374 blv=hd5
hdisk375 blv=hd5
hdisk376 blv=hd5
hdisk377 blv=hd5
```

   In this example, the output should show the default boot volume, `hd5`, for each path.

4. If the `blv` option is not set for a path to the disk, use the `bootlist` command to set it:

```
# bootlist -m normal hdisk374 hdisk375 hdisk376 hdisk377 blv=hd5
```

5. Run the following command to configure DMP on the root disk:

```
# vxdmpadm native enable vgname=rootvg
```

6. Reboot the system. DMP takes control of the SAN boot device to perform load balancing and failover.

7. Verify whether DMP controls the root disk.

```
# vxdmpadm native list vgname=rootvg
```

Example of migrating an internal root disk to a SAN root disk under DMP control

In this example, a SAN boot disk with multiple paths is created by cloning the existing root disk, and then enabling multi-pathing support by DMP.

To migrate an internal root disk to a SAN root disk under DMP control

1. Clear the PVIDs of all the paths to the SAN boot disk. If the SAN disk is under VM control, then you can get multi-pathing information using the `vxdmpadm` command:

   ```
   # vxdmpadm getsubpaths dmpnodename=ams_wms0_1
   ```

```
NAME     STATE[A]   PATH-TYPE[M] CTLR-NAME ENCLR-TYPE ENCLR-NAME ATTRS
=====================================================================
hdisk542 ENABLED(A)  PRIMARY     fscsi0    AMS_WMS    ams_wms0   -
hdisk557 ENABLED     SECONDARY   fscsi0    AMS_WMS    ams_wms0   -
hdisk558 ENABLED(A)  PRIMARY     fscsi1    AMS_WMS    ams_wms0   -
hdisk559 ENABLED     SECONDARY   fscsi1    AMS_WMS    ams_wms0   -
```

   Clear the PVIDs of all these paths.

   ```
   # chdev -l hdisk542 -a pv=clear
   hdisk542 changed
   # chdev -l hdisk557 -a pv=clear
   hdisk557 changed
   # chdev -l hdisk558 -a pv=clear
   hdisk558 changed
   # chdev -l hdisk559 -a pv=clear
   hdisk559 changed
   ```

   Note that unless the disk is under VM control, the clear command may not work for secondary paths.

2 Clone `rootvg` on the SAN disk. If the disk is under VM control, then remove the disk from VM control before proceeding:

```
# vxdiskunsetup ams_wms0_1
```

```
# vxdisk rm ams_wms0_1
```

Use the alt_disk_install command to clone the `rootvg` to the SAN boot disk. You can use any of the paths, but give preference to PRIMARY path.

```
# alt_disk_install -C -P all hdisk542
+-------------------------------------------------------------+
ATTENTION: calling new module /usr/sbin/alt_disk_copy. Please
see the
alt_disk_copy man page and documentation for more details.
Executing command: /usr/sbin/alt_disk_copy -P "all" -d
"hdisk542"
+-------------------------------------------------------------+
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical
volumes.
Creating logical volume alt_hd5.
Creating logical volume alt_hd6.
Creating logical volume alt_hd8.
Creating logical volume alt_hd4.
Creating logical volume alt_hd2.
Creating logical volume alt_hd9var.
Creating logical volume alt_hd3.
Creating logical volume alt_hd1.
Creating logical volume alt_hd10opt.
Creating logical volume alt_lg_dumplv.
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the alternate
file system...
Modifying ODM on cloned disk.
Building boot image on cloned disk.
```

```
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk542
```

3 Use the `lspv` command to confirm that `altinst_rootvg` has been created for one of the paths to the SAN disk:

```
# lspv | grep rootvg
hdisk125 00cdee4fd0e3b3da rootvg active
hdisk542 00cdee4f5b103e98 altinst_rootvg
```

4 Update the remaining paths to the SAN disk to include the correct `altinst_rootvg` information:

```
# chdev -l hdisk557 -a pv=yes
hdisk557 changed
# chdev -l hdisk558 -a pv=yes
hdisk558 changed
# chdev -l hdisk559 -a pv=yes
hdisk559 changed
# lspv | grep rootvg
hdisk125 00cdee4fd0e3b3da rootvg active
hdisk542 00cdee4f5b103e98 altinst_rootvg
hdisk557 00cdee4f5b103e98 altinst_rootvg
hdisk558 00cdee4f5b103e98 altinst_rootvg
hdisk559 00cdee4f5b103e98 altinst_rootvg
```

5 The `bootlist` command verifies that the boot device has been updated for only one of the paths to the SAN disk:

```
# bootlist -m normal -o
hdisk542 blv=hd5
```

6 Use the `bootlist` command to include the other paths to the new boot device:

```
# bootlist -m normal hdisk542 hdisk557 hdisk558 hdisk559 blv=hd5
# bootlist -m normal -o
hdisk542 blv=hd5
hdisk557 blv=hd5
hdisk558 blv=hd5
hdisk559 blv=hd5
```

7 Run the following command to make sure all the disks in step 6 are bootable:

```
# ipl_varyon -i | egrep -w "hdisk542|hdisk557|hdisk558|hdisk559"
hdisk542 YES oocdee4fd0e3b3da00000000000000 00cbf5ce00004c00
hdisk557 YES oocdee4f5b103e980000000000000000 00cbf5ce00004c00
hdisk558 YES oocdee4f5b103e980000000000000000 00cbf5ce00004c00
hdisk559 YES oocdee4f5b103e980000000000000000 00cbf5ce00004c00
```

8 DMP can now be enabled:

```
# vxdmpadm native enable vgname=rootvg
```

9 Reboot the system from the SAN disk.

10 After rebooting the system to enable DMP support for LVM bootablity, confirm that the system is booted from the new multipathed SAN disk. Use the following commands:

```
# bootinfo -b
hdisk542
# bootlist -m normal -o
hdisk542 blv=hd5
hdisk557 blv=hd5
hdisk558 blv=hd5
hdisk559 blv=hd5
# lspv | grep rootvg
hdisk125 00cdee4fd0e3b3da old_rootvg
hdisk542 00cdee4f5b103e98 rootvg active
hdisk557 00cdee4f5b103e98 rootvg active
hdisk558 00cdee4f5b103e98 rootvg active
hdisk559 00cdee4f5b103e98 rootvg active
```

11 Verify whether DMP controls the root disk..

```
# vxdmpadm native list vgname=rootvg
```

Extending a native rootvg that is enabled for DMP

When a native root volume group (rootvg) is enabled for DMP, use this procedure to extend the rootvg by adding a SAN disk.

To add a SAN disk to a DMP-enabled rootvg

1. Clear the physical volume Identifiers (PVIDs) of all the paths to the SAN disk. Perform this step for each of the paths.

   ```
   # vxdmpadm getsubpaths dmpnodename=eva4k6k0_0
   hdisk131  ENABLED(A)  PRIMARY    fscsi0   EVA4K6K  eva4k6k0    -
   hdisk132  ENABLED     SECONDARY  fscsi0   EVA4K6K  eva4k6k0    -
   hdisk133  ENABLED(A)  PRIMARY    fscsi1   EVA4K6K  eva4k6k0    -
   hdisk134  ENABLED     SECONDARY  fscsi1   EVA4K6K  eva4k6k0    -
   ```

   ```
   # chdev -l hdisk131 -a pv=clear
   ```

2. If the disk is under VxVM control, remove the disk from VxVM before you continue.

   ```
   # vxdisk rm eva4k6k0_0
   ```

3. Add the SAN disk to the DMP-enabled rootvg

   ```
   # extendvg rootvg hdisk131
   ```

4. Update the PVID on the remaining paths of added SAN disk. Perform this step for each of the paths.

   ```
   # chdev -l hdisk132 -a pv=yes
   ```

5 Reboot the system.

```
# reboot
```

6 Verify the DMP rootvg configuration.

```
# vxdmpadm native list vgname=rootvg
PATH                          DMPNODENAME
==============================================
hdisk143                      ams_wms0_302
hdisk142                      ams_wms0_302
hdisk141                      ams_wms0_302
hdisk127                      ams_wms0_302
hdisk134                      eva4k6k0_0
hdisk133                      eva4k6k0_0
hdisk132                      eva4k6k0_0
hdisk131                      eva4k6k0_0
```

Reducing the native rootvg that is enabled for DMP

When a native root volume group (rootvg) is enabled for DMP, and contains multiple SAN disks, you can reduce the rootvg. Use this procedure to remove a SAN disk from a rootvg that includes multiple SAN disks.

To remove a SAN disk from a DMP-enabled rootvg

1 Remove the SAN disk from the DMP-enabled rootvg.

```
# lsvg -p rootvg
rootvg:
PV_NAME    PV STATE   TOTAL PPs FREE PPs  FREE DISTRIBUTION
hdisk141   active     319       46        00..00..00..00..46
hdisk131   active     319       319       64..64..63..64..64

# reducevg rootvg hdisk131
```

2 Reboot the system.

```
# reboot
```

3 Verify the DMP rootvg configuration.

```
# vxdmpadm native list vgname=rootvg

PATH                          DMPNODENAME
==============================================
hdisk143                      ams_wms0_302
hdisk142                      ams_wms0_302
hdisk141                      ams_wms0_302
hdisk127                      ams_wms0_302
```

Upgrading Storage Foundation and AIX on a DMP-enabled rootvg

If the rootvg is enabled for DMP, refer to the *Storage Foundation High Availability Installation Guide* for instructions on how to upgrade Storage Foundation, AIX or both.

Upgrading Storage Foundation and AIX on an alternate rootvg that is enabled for DMP on an LVM root disk

To upgrade Storage Foundation and AIX on an alternate rootvg that is enabled for DMP on an LVM root disk, perform the following procedure.

To upgrade Storage Foundation on altinst_rootvg on a DMP-enabled rootvg

1. Clone the currently running system to the alternate disk. For example:

   ```
   # alt_disk_copy -d <hdisk##>
   ```

2. Boot from the alternate disk.
3. Upgrade AIX and Storage Foundation according to the instructions for alternate root upgrade.

   See the *Veritas Storage Foundation High Availability Installation Guide*.
4. Enable DMP on the root disk.

   See "Configuring DMP support for booting over a SAN" on page 78.

Using Storage Foundation in the VIO client with virtual SCSI devices

Storage Foundation provides support for virtual SCSI (vSCSI) devices on the VIO client. You can create and manage Veritas Volume Manager (VxVM) volumes on vSCSI devices, as for any other devices. Storage Foundation provides Dynamic Multi-Pathing (DMP) for vSCSI devices, by default. Storage Foundation can also co-exist with MPIO for multi-pathing. If you choose to use MPIO to multipath the vSCSI devices, DMP works in pass-through mode.

Use the `vxddladm` utility and the `vxdmpadm` utility to administer DMP for vSCSI devices. The `vxddladm` utility controls enabling and disabling DMP on vSCSI devices, adding and removing supported arrays, and listing supported arrays. The `vxdmpadm` utility controls the I/O policy and the path policy for vSCSI devices.

Setting up DMP for vSCSI devices in the Virtual I/O Client

In this release of Storage Foundation, Veritas Dynamic Multi-Pathing (DMP) is enabled on VIO clients by default. After you install or upgrade Storage Foundation in the Virtual IO client, any vSCSI devices are under DMP control. MPIO is disabled.

If you have already installed or upgraded Storage Foundation in the Virtual I/O client, use the following procedure to enable DMP support for vSCSI devices. This procedure is only required if you have previously disabled DMP support for vSCSI devices.

To enable vSCSI support within DMP and disable MPIO

1. Enable vSCSI support.

   ```
   # vxddladm enablevscsi
   ```

2. You are prompted to reboot the devices, if required.

DMP takes control of the devices, for any array that has DMP support to use the array for vSCSI devices. You can add or remove DMP support for vSCSI for arrays.

See "Adding and removing DMP support for vSCSI devices for an array" on page 90.

About disabling DMP multi-pathing for vSCSI devices in the Virtual IO Client

Storage Foundation can co-exist with MPIO multi-pathing in the Virtual I/O client. If you prefer to use MPIO for multi-pathing, you can override the default behavior, which enables Dynamic Multi-Pathing (DMP) in the Virtual I/O client.

There are two ways to do this:

- Before you install or upgrade Storage Foundation in the Virtual I/O client
  See "Preparing to install or upgrade Storage Foundation with DMP disabled for vSCSI devices in the Virtual I/O client" on page 89.
- After Storage Foundation is installed in the Virtual I/O client
  See "Disabling DMP multi-pathing for vSCSI devices in the Virtual IO Client, after installation" on page 90.

Preparing to install or upgrade Storage Foundation with DMP disabled for vSCSI devices in the Virtual I/O client

Before you install or upgrade Storage Foundation, you can set an environment variable to disable DMP use for the vSCSI devices. Storage Foundation is installed with DMP in pass-through mode. MPIO is enabled for multi-pathing.

Note: When you upgrade an existing VxVM installation that has DMP enabled, then DMP remains enabled regardless of whether or not the environment variable __VXVM_DMP_VSCSI_ENABLE is set to no.

To disable DMP before installing or upgrading SF in the Virtual I/O Client

1. Before you install or upgrade VxVM, set the environment variable __VXVM_DMP_VSCSI_ENABLE to no.

   ```
   # export __VXVM_DMP_VSCSI_ENABLE=no
   ```

   Note: The environment variable name __VXVM_DMP_VSCSI_ENABLE begins with two underscore (_) characters.

   ---

2. Install Storage Foundation, as described in the *Storage Foundation High Availability Installation Guide*

Disabling DMP multi-pathing for vSCSI devices in the Virtual IO Client, after installation

After VxVM is installed, use the `vxddladm` command to switch vSCSI devices between MPIO control and DMP control.

To return control to MPIO, disable vSCSI support with DMP. After DMP support has been disabled, MPIO takes control of the devices. MPIO implements multi-pathing features such as failover and load balancing; DMP acts in pass-through mode.

To disable vSCSI support within DMP and enable MPIO

1. Disable vSCSI support.

   ```
   # vxddladm disablevscsi
   ```

2. You are prompted to reboot the devices, if required.

Adding and removing DMP support for vSCSI devices for an array

Veritas Dynamic Multi-Pathing (DMP) controls the devices for any array that has DMP support to use the array for vSCSI devices.

To add or remove DMP support for an array for use with vSCSI devices

1. To determine if DMP support is enabled for an array, list all of the arrays that DMP supports for use with vSCSI devices:

   ```
   # vxddladm listvscsi
   ```

2. If the support is not enabled, add support for using an array as a vSCSI device within DMP:

   ```
   # vxddladm addvscsi array_vid
   ```

3. If the support is enabled, you can remove the support so that the array is not used for vSCSI devices within DMP:

   ```
   # vxddladm rmvscsi array_vid
   ```

4. You are prompted to reboot the system, if required.

How DMP handles I/O for vSCSI devices

On the VIO client, DMP uses the Active/Standby array mode for the vSCSI devices. Each path to the vSCSI device is through a VIO server. One VIO server is Active and the other VIO servers are Standby. An Active/Standby array permits I/O through a single Active path, and keeps the other paths on standby. During failover, I/O is scheduled on one of the standby paths. After failback, I/Os are scheduled back onto the original Active path. The Active/Standby mode is a variation of an active/active array; only one path is active at a time.

The DMP I/O policy for vSCSI devices is always Single-Active. You cannot change the DMP I/O policy for the vSCSI enclosure. Because only one VIO server can be Active, DMP cannot do I/O balancing across the paths for vSCSI devices.

The following command shows the vSCSI enclosure:

```
# vxdmpadm listenclosure all
ENCLR_NAME   ENCLR_TYPE  ENCLR_SNO   STATUS     ARRAY_TYPE  LUN_COUNT
=====================================================================
ibm_vscsi0   IBM_VSCSI   VSCSI       CONNECTED  VSCSI    9
```

The following command shows the I/O policy for the vSCSI enclosure:

```
# vxdmpadm getattr enclosure ibm_vscsi0 iopolicy
ENCLR_NAME     DEFAULT        CURRENT
============================================
ibm_vscsi0     Single-Active  Single-Active
```

For vSCSI devices, DMP balances the load between the VIO servers, instead of balancing the I/O on paths. By default, the `iopolicy` attribute of the vSCSI enclosure is set to `lunbalance`. When `lunbalance` is set, the vSCSI LUNs are distributed so that the I/O load is shared across the VIO servers. For example, if you have 10 LUNs and 2 VIO servers, 5 of them are configured so that VIO Server 1 is Active and VIO Server 2 is Standby. The other 5 are configuredd so that the VIO Server 2 is Active and VIO Server 1 is Standby. To turn off load sharing across VIO servers, set the `iopolicy` attribute to `nolunbalance`.

DMP dynamically balances the I/O load across LUNs. When you add or remove disks or paths in the VIO client, the load is rebalanced. Temporary failures like enabling or disabling paths or controllers do not cause the I/O load across LUNs to be rebalanced.

Setting the vSCSI I/O policy

By default, DMP balances the I/O load across VIO servers. This behavior sets the I/O policy attribute to `lunbalance`.

To display the current I/O policy attribute for the vSCSI array

◆ Display the current I/O policy for a vSCSI array:

```
# vxdmpadm getattr vscsi iopolicy
VSCSI     DEFAULT      CURRENT
============================================
IOPolicy  lunbalance   lunbalance
```

To turn off the LUN balancing, set the I/O policy attribute for the vSCSI array to `nolunbalance`.

To set the I/O policy attribute for the vSCSI array

◆ Set the I/O policy for a vSCSI array:

```
# vxdmpadm setattr vscsi iopolicy={lunbalance|nolunbalance}
```

Note: The DMP I/O policy for each vSCSI device is always Single-Active. You cannot change the DMP I/O policy for the vSCSI enclosure. Only one VIO server can be Active for each vSCSI device.

Running alt_disk_install, alt_disk_copy and related commands on the OS device when DMP native support is enabled

When DMP is enabled for native OS devices, you can use the following procedures to run the `alt_disk_install` command, `alt_disk_copy` command, or related commands on the operating system device.

Running alt_disk_install in the physical environment

1 Find the DMP device corresponding to the OS device path on which you plan to run the `alt_disk_install` command.

```
# vxdmpadm getdmpnode nodename=hdisk13
NAME        STATE    ENCLR-TYPE PATHS ENBL DSBL ENCLR-NAME
==========================================================
emc0_0039   ENABLED  EMC        4     4    0    emc0
```

2 Close references to the associated subpaths. Run the following command on the DMP device:

```
# vxdisk rm emc0_0039
```

3 Run the `alt_disk_install` command on the OS device.

Refer to the OS vendor documentation for the `alt_disk_install` command.

Running alt_disk_install in the VIOS environment

1 Find the DMP device corresponding to the OS device path on which you plan to run the `alt_disk_install` command.

```
# vxdmpadm getdmpnode nodename=hdisk13
NAME        STATE    ENCLR-TYPE PATHS ENBL DSBL ENCLR-NAME
==========================================================
emc0_0039   ENABLED  EMC        4     4    0    emc0
```

2 If the DMP device is exported to a VIO client, remove the mapping of the DMP device. From the VIOS, run the following command:

```
# /usr/ios/cli/ioscli rmvdev -vtd VTD_devicename
```

3 Close references to the associated subpaths. Run the following command on the DMP device:

```
# vxdisk rm emc0_0039
```

4 Run the `alt_disk_install` command on the OS device.

Refer to the OS vendor documentation for the `alt_disk_install` command.

Administering DMP using vxdmpadm

The `vxdmpadm` utility is a command line administrative interface to DMP.

You can use the `vxdmpadm` utility to perform the following tasks:

- Retrieve the name of the DMP device corresponding to a particular path.
- Display the members of a LUN group.
- List all paths under a DMP device node, HBA controller or array port.
- Display information about the HBA controllers on the host.
- Display information about enclosures.
- Display information about array ports that are connected to the storage processors of enclosures.
- Display information about devices that are controlled by third-party multi-pathing drivers.
- Gather I/O statistics for a DMP node, enclosure, path or controller.
- Configure the attributes of the paths to an enclosure.
- Set the I/O policy that is used for the paths to an enclosure.
- Enable or disable I/O for a path, HBA controller or array port on the system.
- Upgrade disk controller firmware.
- Rename an enclosure.
- Configure how DMP responds to I/O request failures.
- Configure the I/O throttling mechanism.
- Control the operation of the DMP path restoration thread.
- Get or set the values of various tunables used by DMP.

The following sections cover these tasks in detail along with sample output.

See “DMP tunable parameters” on page 170.

See the vxdmpadm(1M) manual page.

Retrieving information about a DMP node

The following command displays the DMP node that controls a particular physical path:

```
# vxdmpadm getdmpnode nodename=hdisk107
```

The physical path is specified by argument to the nodename attribute, which must be a valid path listed in the /dev directory.

The command displays output similar to the following:

```
NAME             STATE   ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
===================================================================
emc_clariion0_17 ENABLED EMC_CLARiiON 8     8    0    emc_clariion0
```

Use the -v option to display the LUN serial number and the array volume ID.

```
# vxdmpadm -v getdmpnode nodename=hdisk107
```

```
NAME             STATE   ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME    SERIAL-NO ARRAY_VOL_ID
=============================================================================================
emc_clariion0_17 ENABLED EMC_CLARiiON 8     8    0    emc_clariion0 600601601 17
```

Use the enclosure attribute with getdmpnode to obtain a list of all DMP nodes for the specified enclosure.

```
# vxdmpadm getdmpnode enclosure=enc0
```

```
NAME     STATE    ENCLR-TYPE PATHS  ENBL  DSBL ENCLR-NAME
===========================================================
hdisk11  ENABLED  ACME       2      2     0    enc0
hdisk12  ENABLED  ACME       2      2     0    enc0
hdisk13  ENABLED  ACME       2      2     0    enc0
hdisk14  ENABLED  ACME       2      2     0    enc0
```

Use the dmpnodename attribute with getdmpnode to display the DMP information for a given DMP node.

```
# vxdmpadm getdmpnode dmpnodename=emc_clariion0_158
```

```
NAME              STATE   ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
==================================================================
emc_clariion0_158 ENABLED EMC_CLARiiON 1     1    0    emc_clariion0
```

Displaying consolidated information about the DMP nodes

The `vxdmpadm list dmpnode` command displays the detail information of a DMP node. The information includes the enclosure name, LUN serial number, port id information, device attributes, etc.

The following command displays the consolidated information for all of the DMP nodes in the system:

```
# vxdmpadm list dmpnode all
```

Use the `enclosure` attribute with `list dmpnode` to obtain a list of all DMP nodes for the specified enclosure.

```
# vxdmpadm list dmpnode enclosure=enclosure name
```

For example, the following command displays the consolidated information for all of the DMP nodes in the `enc0` enclosure.

```
#vxdmpadm list dmpnode enclosure=enc0
```

Use the `dmpnodename` attribute with `list dmpnode` to display the DMP information for a given DMP node. The DMP node can be specified by name or by specifying a path name. The detailed information for the specified DMP node includes path information for each subpath of the listed dmpnode.

The path state differentiates between a path that is disabled due to a failure and a path that has been manually disabled for administrative purposes. A path that has been manually disabled using the `vxdmpadm disable` command is listed as `disabled(m)`.

```
# vxdmpadm list dmpnode dmpnodename=dmpnodename
```

For example, the following command displays the consolidated information for the DMP node emc_clariion0_158.

```
# vxdmpadm list dmpnode dmpnodename=emc_clariion0_158

dmpdev      = emc_clariion0_19
state       = enabled
enclosure   = emc_clariion0
cab-sno     = APM00042102192
asl         = libvxCLARiiON.so
vid         = DGC
pid         = CLARiiON
array-name  = EMC_CLARiiON
array-type  = CLR-A/P
```

```
iopolicy     = MinimumQ
avid         = -
lun-sno      = 6006016070071100F6BF98A778EDD811
udid         = DGC%5FCLARiiON%5FAPM00042102192%5F6006016070071100F6BF98A778EDD811
dev-attr     = -
###path      = name state type transport ctlr hwpath aportID aportWWN attr
path         = hdisk11 enabled(a) primary FC fscsi0 07-08-02 B0APM00042102192
50:06:01:68:10:21:26:c1 -
path         = hdisk31 disabled secondary FC fscsi1 08-08-02 A0APM00042102192
50:06:01:60:10:21:26:c1 -
```

Displaying the members of a LUN group

The following command displays the DMP nodes that are in the same LUN group as a specified DMP node:

```
# vxdmpadm getlungroup dmpnodename=hdisk16

NAME      STATE     ENCLR-TYPE  PATHS      ENBL     DSBL    ENCLR-NAME
===============================================================
hdisk14   ENABLED   ACME         2          2        0        enc1
hdisk15   ENABLED   ACME         2          2        0        enc1
hdisk16   ENABLED   ACME         2          2        0        enc1
hdisk17   ENABLED   ACME         2          2        0        enc1
```

Displaying paths controlled by a DMP node, controller, enclosure, or array port

The `vxdmpadm getsubpaths` command lists all of the paths known to DMP. The `vxdmpadm getsubpaths` command also provides options to list the subpaths through a particular DMP node, controller, enclosure, or array port. To list the paths through an array port, specify either a combination of enclosure name and array port id, or array port WWN. You can also display paths for devices controlled by third-party drivers.

See “Displaying information about TPD-controlled devices” on page 102.

To list all subpaths known to DMP:

```
# vxdmpadm getsubpaths

NAME      STATE[A]   PATH-TYPE[M] DMPNODENAME  ENCLR-NAME   CTLR   ATTRS
=============================================================================
hdisk1    ENABLED(A)   -          disk_0       disk         scsi0    -
hdisk0    ENABLED(A)   -          disk_1       disk         scsi0    -
```

```
hdisk107  ENABLED(A) PRIMARY      emc_clariion0_17 emc_clariion0 fscsi1   -
hdisk17   ENABLED    SECONDARY    emc_clariion0_17 emc_clariion0 fscsi0   -
hdisk108  ENABLED(A) PRIMARY      emc_clariion0_74 emc_clariion0 fscsi1   -
hdisk18   ENABLED    SECONDARY    emc_clariion0_74 emc_clariion0 fscsi0   -
hdisk109  ENABLED(A) PRIMARY      emc_clariion0_75 emc_clariion0 fscsi1   -
hdisk19   ENABLED    SECONDARY    emc_clariion0_75 emc_clariion0 fscsi0   -
```

The `vxdmpadm getsubpaths` command combined with the `dmpnodename` attribute displays all the paths to a LUN that are controlled by the specified DMP node name from the `/dev/vx/rdmp` directory:

```
# vxdmpadm getsubpaths dmpnodename=hdisk22

NAME    STATE[A]   PATH-TYPE[M] CTLR-NAME ENCLR-TYPE ENCLR-NAME ATTRS
====================================================================
hdisk22 ENABLED(A)  PRIMARY     scsi2     ACME       enc0       -
hdisk21 ENABLED     PRIMARY     scsi1     ACME       enc0       -
```

For A/A arrays, all enabled paths that are available for I/O are shown as `ENABLED(A)`.

For A/P arrays in which the I/O policy is set to `singleactive`, only one path is shown as `ENABLED(A)`. The other paths are enabled but not available for I/O. If the I/O policy is not set to `singleactive`, DMP can use a group of paths (all primary or all secondary) for I/O, which are shown as `ENABLED(A)`.

See "Specifying the I/O policy" on page 115.

Paths that are in the DISABLED state are not available for I/O operations.

A path that was manually disabled by the system administrator displays as DISABLED(M). A path that failed displays as DISABLED.

You can use `getsubpaths` to obtain information about all the paths that are connected to a particular HBA controller:

```
# vxdmpadm getsubpaths ctlr=fscsi1

NAME     STATE[A]   PATH-TYPE[M] DMPNODENAME  ENCLR-TYPE   ENCLR-NAME   ATTRS
=============================================================================
hdisk107 ENABLED(A) PRIMARY      emc_clariion0_17 EMC_CLARiiON emc_clariion0 -
hdisk62  ENABLED    SECONDARY    emc_clariion0_17 EMC_CLARiiON emc_clariion0 -
hdisk108 ENABLED(A) PRIMARY      emc_clariion0_74 EMC_CLARiiON emc_clariion0 -
hdisk63  ENABED     SECONDARY    emc_clariion0_74 EMC_CLARiiON emc_clariion0 -
```

You can also use `getsubpaths` to obtain information about all the paths that are connected to a port on an array. The array port can be specified by the name of the enclosure and the array port ID, or by the worldwide name (WWN) identifier of the array port:

```
# vxdmpadm getsubpaths enclosure=enclosure portid=portid
# vxdmpadm getsubpaths pwwn=pwwn
```

For example, to list subpaths through an array port through the enclosure and the array port ID:

```
# vxdmpadm getsubpaths enclosure=emc_clariion0 portid=A2

NAME     STATE[A]   PATH-TYPE[M] DMPNODENAME   ENCLR-NAME   CTLR    ATTRS
========================================================================
hdisk111 ENABLED(A) PRIMARY      emc_clariion0_80 emc_clariion0 fscsi1 -
hdisk51  ENABLED(A) PRIMARY      emc_clariion0_80 emc_clariion0 fscsi0 -
hdisk112 ENABLED(A) PRIMARY      emc_clariion0_81 emc_clariion0 fscsi1 -
hdisk52  ENABLED(A) PRIMARY      emc_clariion0_81 emc_clariion0 fscsi0 -
```

For example, to list subpaths through an array port through the WWN:

```
NAME     STATE[A]   PATH-TYPE[M] DMPNODENAME   ENCLR-NAME   CTLR    ATTRS
========================================================================
hdisk111 ENABLED(A) PRIMARY      emc_clariion0_80 emc_clariion0 fscsi1 -
hdisk51  ENABLED(A) PRIMARY      emc_clariion0_80 emc_clariion0 fscsi0 -
hdisk112 ENABLED(A) PRIMARY      emc_clariion0_81 emc_clariion0 fscsi1 -
hdisk52  ENABLED(A) PRIMARY      emc_clariion0_81 emc_clariion0 fscsi0 -
```

You can use `getsubpaths` to obtain information about all the subpaths of an enclosure.

```
# vxdmpadm getsubpaths enclosure=enclosure_name [ctlr=ctlrname]
```

To list all subpaths of an enclosure:

```
# vxdmpadm getsubpaths enclosure=emc_clariion0
NAME         STATE[A]   PATH-TYPE[M] DMPNODENAME  ENCLR-NAME   CTLR    ATTRS
================================================================================
hdisk107     ENABLED(A) PRIMARY      emc_clariion0_17 emc_clariion0 fscsi1    -
hdisk17      ENABLED    SECONDARY    emc_clariion0_17 emc_clariion0 fscsi0    -
hdisk110     ENABLED(A) PRIMARY      emc_clariion0_76 emc_clariion0 fscsi1    -
hdisk20      ENABLED    SECONDARY    emc_clariion0_76 emc_clariion0 fscsi0    -
```

To list all subpaths of a controller on an enclosure:

```
# vxdmpadm getsubpaths enclosure=Disk ctlr=c1
```

By default, the output of the `vxdmpadm getsubpaths` command is sorted by enclosure name, DMP node name, and within that, path name. To sort the output based on the pathname, the DMP node name, the enclosure name, or the host controller name, use the `-s` option.

To sort subpaths information, use the following command:

```
# vxdmpadm -s {path | dmpnode | enclosure | ctlr} getsubpaths \
[all | ctlr=ctlr_name | dmpnodename=dmp_device_name | \
enclosure=enclr_name [ctlr=ctlr_name | portid=array_port_ID] | \
pwwn=port_WWN | tpdnodename=tpd_node_name]
```

Displaying information about controllers

The following command lists attributes of all HBA controllers on the system:

```
# vxdmpadm listctlr all
```

```
CTLR-NAME       ENCLR-TYPE        STATE         ENCLR-NAME
===============================================================
scsi1           OTHER             ENABLED       other0
scsi2           X1                ENABLED       jbod0
scsi3           ACME              ENABLED       enc0
scsi4           ACME              ENABLED       enc0
```

This output shows that the controller `scsi1` is connected to disks that are not in any recognized DMP category as the enclosure type is `OTHER`.

The other controllers are connected to disks that are in recognized DMP categories.

All the controllers are in the ENABLED state which indicates that they are available for I/O operations.

The state DISABLED is used to indicate that controllers are unavailable for I/O operations. The unavailability can be due to a hardware failure or due to I/O operations being disabled on that controller by using the `vxdmpadm disable` command.

The following forms of the command lists controllers belonging to a specified enclosure or enclosure type:

```
# vxdmpadm listctlr enclosure=enc0
```

or

```
# vxdmpadm listctlr type=ACME
```

```
CTLR-NAME         ENCLR-TYPE          STATE        ENCLR-NAME
================================================================
scsi2             ACME                ENABLED      enc0
scsi3             ACME                ENABLED      enc0
```

The `vxdmpadm getctlr` command displays HBA vendor details and the Controller ID. For iSCSI devices, the Controller ID is the IQN or IEEE-format based name. For FC devices, the Controller ID is the WWN. Because the WWN is obtained from ESD, this field is blank if ESD is not running. ESD is a daemon process used to notify DDL about occurance of events. The WWN shown as 'Controller ID' maps to the WWN of the HBA port associated with the host controller.

```
# vxdmpadm getctlr fscsi2
```

```
LNAME     PNAME       VENDOR    CTLR-ID
===============================================================
fscsi2    20-60-01    IBM       10:00:00:00:c9:2d:26:11
```

Displaying information about enclosures

To display the attributes of a specified enclosure, including its enclosure type, enclosure serial number, status, array type, and number of LUNs, use the following command:

```
# vxdmpadm listenclosure enc0
```

```
ENCLR_NAME ENCLR_TYPE  ENCLR_SNO            STATUS   ARRAY_TYPE    LUN_COUNT
==============================================================================
enc0       A3          60020f20000001a90000 CONNECTED  A/P         30
```

The following command lists attributes for all enclosures in a system:

```
# vxdmpadm listenclosure all
```

```
ENCLR_NAME ENCLR_TYPE ENCLR_SNO            STATUS    ARRAY_TYPE    LUN_COUNT
==================================================================================
Disk       Disk       DISKS                CONNECTED     Disk        6
ANA0       ACME       508002000001d660     CONNECTED     A/A        57
enc0       A3         60020f20000001a90000 CONNECTED     A/P        30
```

If an A/P or ALUA array is under the control of MPIO, then DMP claims the devices in A/A mode. The output of the above commands shows the ARRAY_TYPE as A/A. For arrays under MPIO control, DMP does not store A/P-specific attributes or ALUA-specific attributes. These attributes include primary/secondary paths, port serial number, and the array controller ID.

Displaying information about array ports

Use the commands in this section to display information about array ports. The information displayed for an array port includes the name of its enclosure, and its ID and worldwide name (WWN) identifier.

Note: DMP does not report information about array ports for LUNs that are controlled by the native multi-pathing driver..DMP reports pWWN information only if the dmp_monitor_fabric tunable is on, and the event source daemon (esd) is running.

To display the attributes of an array port that is accessible via a path, DMP node or HBA controller, use one of the following commands:

```
# vxdmpadm getportids path=path-name
# vxdmpadm getportids dmpnodename=dmpnode-name
# vxdmpadm getportids ctlr=ctlr-name
```

The following form of the command displays information about all of the array ports within the specified enclosure:

```
# vxdmpadm getportids enclosure=enclr-name
```

The following example shows information about the array port that is accessible via DMP node `hdisk12`:

```
# vxdmpadm getportids dmpnodename=hdisk12

NAME        ENCLR-NAME  ARRAY-PORT-ID pWWN
==============================================================
hdisk12     HDS9500V0   1A            20:00:00:E0:8B:06:5F:19
```

Displaying information about TPD-controlled devices

The third-party driver (TPD) coexistence feature allows I/O that is controlled by third-party multi-pathing drivers to bypass DMP while retaining the monitoring capabilities of DMP. The following commands allow you to display the paths that DMP has discovered for a given TPD device, and the TPD device that corresponds to a given TPD-controlled node discovered by DMP:

```
# vxdmpadm getsubpaths tpdnodename=TPD_node_name
# vxdmpadm gettpdnode nodename=TPD_path_name
```

For example, consider the following disks in an EMC Symmetrix array controlled by PowerPath, which are known to DMP:

```
# vxdisk list

DEVICE          TYPE            DISK       GROUP     STATUS
hdiskpower10    auto:cdsdisk    disk1      ppdg      online
hdiskpower11    auto:cdsdisk    disk2      ppdg      online
hdiskpower12    auto:cdsdisk    disk3      ppdg      online
hdiskpower13    auto:cdsdisk    disk4      ppdg      online
hdiskpower14    auto:cdsdisk    disk5      ppdg      online
hdiskpower15    auto:cdsdisk    disk6      ppdg      online
hdiskpower16    auto:cdsdisk    disk7      ppdg      online
hdiskpower17    auto:cdsdisk    disk8      ppdg      online
hdiskpower18    auto:cdsdisk    disk9      ppdg      online
hdiskpower19    auto:cdsdisk   disk10     ppdg      online
```

The following command displays the paths that DMP has discovered, and which correspond to the PowerPath-controlled node, `emcpower10`:

```
# vxdmpadm getsubpaths tpdnodename=hdiskpower10

NAME    TPDNODENAME   PATH-TYPE[-]DMP-NODENAME   ENCLR-TYPE ENCLR-NAME
====================================================================
hdisk10 hdiskpower10s2 -           hdiskpower10    EMC       EMC0
hdisk20 hdiskpower10s2 -           hdiskpower10    EMC       EMC0
```

Conversely, the next command displays information about the PowerPath node that corresponds to the path, hdisk10, discovered by DMP:

```
# vxdmpadm gettpdnode nodename=hdiskpower10

NAME          STATE    PATHS  ENCLR-TYPE    ENCLR-NAME
====================================================================
hdiskpower10s2 ENABLED  2      EMC           EMC0
```

Displaying extended device attributes

Device Discovery Layer (DDL) extended attributes are attributes or flags corresponding to a VxVM or DMP LUN or Disk and which are discovered by DDL. These attributes identify a LUN to a specific hardware category.

The list of categories includes:

Hardware RAID types	Displays what kind of Storage RAID Group the LUN belongs to

Thin Provisioning Discovery and Reclamation	Displays the LUN's thin reclamation abilities
Device Media Type	Displays the type of media –whether SSD (solid state disk)
Storage-based Snapshot/Clone	Displays whether the LUN is a SNAPSHOT or a CLONE of a PRIMARY LUN
Storage-based replication	Displays if the LUN is part of a replicated group across a remote site
Transport	Displays what kind of HBA is used to connect to this LUN (FC, SATA, iSCSI)

Each LUN can have one or more of these attributes discovered during device discovery. ASLs furnish this information to DDL through the property DDL_DEVICE_ATTR. The `vxdisk -p list` command displays DDL extended attributes. For example, the following command shows attributes of "std", "fc", and "RAID_5" for this LUN:

```
# vxdisk -p list
DISK           : tagmastore-usp0_0e18
DISKID         : 1253585985.692.rx2600h11
VID            : HITACHI
UDID           : HITACHI%5FOPEN-V%5F02742%5F0E18
REVISION       : 5001
PID            : OPEN-V
PHYS_CTLR_NAME : 0/4/1/1.0x50060e8005274246
LUN_SNO_ORDER  : 411
LUN_SERIAL_NO  : 0E18
LIBNAME        : libvxhdsusp.sl
HARDWARE_MIRROR: no
DMP_DEVICE     : tagmastore-usp0_0e18
DDL_THIN_DISK  : thick
DDL_DEVICE_ATTR: std fc RAID_5
CAB_SERIAL_NO  : 02742
ATYPE          : A/A
ARRAY_VOLUME_ID: 0E18
ARRAY_PORT_PWWN: 50:06:0e:80:05:27:42:46
ANAME          : TagmaStore-USP
TRANSPORT      : FC
```

The `vxdisk -x attribute -p list` command displays the one-line listing for the property list and the attributes. The following example shows two Hitachi LUNs that support Thin Reclamation via the attribute `hdprclm`:

```
# vxdisk -x DDL_DEVICE_ATTR -p list
DEVICE                    DDL_DEVICE_ATTR
tagmastore-usp0_0a7a      std fc RAID_5
tagmastore-usp0_065a      hdprclm fc
tagmastore-usp0_065b      hdprclm fc
```

User can specify multiple `-x` options in the same command to display multiple entries. For example:

```
# vxdisk -x DDL_DEVICE_ATTR -x VID -p list

DEVICE               VID          DDL_DEVICE_ATTR
tagmastore-usp0_0a7a HITACHI      std fc RAID_5
tagmastore-usp0_0a7b HITACHI      std fc RAID_5
tagmastore-usp0_0a78 HITACHI      std fc RAID_5
tagmastore-usp0_0a79 HITACHI      std fc RAID_5
tagmastore-usp0_065a HITACHI      hdprclm fc
tagmastore-usp0_065b HITACHI      hdprclm fc
tagmastore-usp0_065c HITACHI      hdprclm fc
tagmastore-usp0_065d HITACHI      hdprclm fc
```

Use the `vxdisk -e list` command to show the DLL_DEVICE_ATTR property in the last column named ATTR.

```
# vxdisk -e list
DEVICE               TYPE  DISK   GROUP  STATUS  OS_NATIVE_NAME  ATTR
tagmastore-usp0_0a7a auto  -      -      online  c10t0d2         std fc RAID_5
tagmastore-usp0_0a7b auto  -      -      online  c10t0d3         std fc RAID_5
tagmastore-usp0_0a78 auto  -      -      online  c10t0d0         std fc RAID_5
tagmastore-usp0_0655 auto  -      -      online  c13t2d7         hdprclm fc
tagmastore-usp0_0656 auto  -      -      online  c13t3d0         hdprclm fc
tagmastore-usp0_0657 auto  -      -      online  c13t3d1         hdprclm fc
```

For a list of ASLs that supports Extended Attributes, and descriptions of these attributes, refer to the hardware compatibility list at the following URL:

http://seer.entsupport.symantec.com/docs/330441.htm

Note: DMP does not support Extended Attributes for LUNs that are controlled by the native multi-pathing driver.

Suppressing or including devices for VxVM or DMP control

The `vxdmpadm exclude` command suppresses devices from VxVM or DMP based on the criteria that you specify. The devices can be added back into VxVM or DMP control by using the `vxdmpadm include` command. The devices can be included or excluded based on VID:PID combination, paths, controllers, or disks. You can use the bang symbol (!) to exclude or include any paths or controllers except the one specified.

The root disk cannot be suppressed.The operation fails if the VID:PID of an external disk is the same VID:PID as the root disk and the root disk is under DMP rootability control.

Note: The ! character is a special character in some shells. The following syntax shows how to escape it in a bash shell.

```
# vxdmpadm exclude [vxvm | vxdmp] { all | product=VID:PID |
ctlr=[\!]ctlr | dmpnodename=diskname [ path=\!pathname] }
```

```
# vxdmpadm include [vxvm | vxdmp] { all | product=VID:PID |
ctlr=[\!]ctlr | dmpnodename=diskname [ path=\!pathname] }
```

where:

all – all devices

product=*VID:PID* – all devices with the specified VID:PID

ctlr=*ctlr* – all devices through the given controller

dmpnodename=*diskname* - all paths under the DMP node

dmpnodename=*diskname* path=\!*pathname* - all paths under the DMP node except the one specified.

Gathering and displaying I/O statistics

You can use the `vxdmpadm iostat` command to gather and display I/O statistics for a specified DMP node, enclosure, path or controller.

To enable the gathering of statistics, enter this command:

```
# vxdmpadm iostat start [memory=size]
```

To reset the I/O counters to zero, use this command:

```
# vxdmpadm iostat reset
```

The `memory` attribute can be used to limit the maximum amount of memory that is used to record I/O statistics for each CPU. The default limit is `32k` (32 kilobytes) per CPU.

To display the accumulated statistics at regular intervals, use the following command:

```
# vxdmpadm iostat show {all | ctlr=ctlr-name \
  | dmpnodename=dmp-node  \
  | enclosure=enclr-name [portid=array-portid ] \
  | pathname=path-name |  pwwn=array-port-wwn } \
  [interval=seconds [count=N]]
```

This command displays I/O statistics for all paths (`all`), or for a specified controller, DMP node, enclosure, path or port ID. The statistics displayed are the CPU usage and amount of memory per CPU used to accumulate statistics, the number of read and write operations, the number of kilobytes read and written, and the average time in milliseconds per kilobyte that is read or written.

The `interval` and `count` attributes may be used to specify the interval in seconds between displaying the I/O statistics, and the number of lines to be displayed. The actual interval may be smaller than the value specified if insufficient memory is available to record the statistics.

To disable the gathering of statistics, enter this command:

```
# vxdmpadm iostat stop
```

Examples of using the vxdmpadm iostat command

The following is an example session using the `vxdmpadm iostat` command. The first command enables the gathering of I/O statistics:

```
# vxdmpadm iostat start
```

The next command displays the current statistics including the accumulated total numbers of read and write operations and kilobytes read and written, on all paths.

```
# vxdmpadm iostat show all
                       cpu usage = 7952us    per cpu memory = 8192b
                OPERATIONS           KBYTES              AVG TIME(ms)
PATHNAME    READS      WRITES     READS     WRITES      READS   WRITES
hdisk10        87           0     44544          0       0.00     0.00
hdisk16         0           0         0          0       0.00     0.00
hdisk11        87           0     44544          0       0.00     0.00
hdisk17         0           0         0          0       0.00     0.00
```

```
hdisk12        87           0     44544           0       0.00      0.00
hdisk18         0           0         0           0       0.00      0.00
hdisk13        87           0     44544           0       0.00      0.00
hdisk19         0           0         0           0       0.00      0.00
hdisk14        87           0     44544           0       0.00      0.00
hdisk20         0           0         0           0       0.00      0.00
hdisk15        87           0     44544           0       0.00      0.00
hdisk21         0           0         0           0       0.00      0.00
```

The following command changes the amount of memory that `vxdmpadm` can use to accumulate the statistics:

```
# vxdmpadm iostat start memory=4096
```

The displayed statistics can be filtered by path name, DMP node name, and enclosure name (note that the per-CPU memory has changed following the previous command):

```
# vxdmpadm iostat show pathname=hdisk17
                      cpu usage = 8132us    per cpu memory = 4096b
                OPERATIONS             KBYTES             AVG TIME(ms)
PATHNAME     READS      WRITES       READS      WRITES     READS   WRITES
hdisk17          0           0           0           0      0.00     0.00

# vxdmpadm iostat show dmpnodename=hdisk10
                     cpu usage = 8501us    per cpu memory = 4096b
               OPERATIONS            KBYTES             AVG  TIME(ms)
PATHNAME    READS      WRITES       READS      WRITES      READS   WRITES
hdisk10      1088           0     557056           0       0.00     0.00

# vxdmpadm iostat show enclosure=Disk
                     cpu usage = 8626us    per cpu memory = 4096b
               OPERATIONS            KBYTES             AVG TIME(ms)
PATHNAME    READS      WRITES       READS      WRITES      READS   WRITES
hdisk10      1088           0     557056           0       0.00     0.00
```

You can also specify the number of times to display the statistics and the time interval. Here the incremental statistics for a path are displayed twice with a 2-second interval:

```
# vxdmpadm iostat show pathname=hdisk17 interval=2 count=2
                      cpu usage = 719us    per cpu memory = 49152b
               OPERATIONS            BLOCKS            AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES      READS     WRITES
hdisk17     0         0           0          0            0.00       0.00
```

```
hdisk17    0         0          0          0          0.00      0.00
```

Displaying statistics for queued or erroneous I/Os

Use the `vxdmpadm iostat show` command with the `-q` option to display the I/Os queued in DMP for a specified DMP node, or for a specified path or controller. For a DMP node, the `-q` option displays the I/Os on the specified DMP node that were sent to underlying layers. If a path or controller is specified, the `-q` option displays I/Os that were sent to the given path or controller and not yet returned to DMP.

See the `vxdmpadm`(1m) manual page for more information about the `vxdmpadm iostat` command.

To display queued I/O counts on a DMP node:

```
# vxdmpadm -q iostat show [filter]
[interval=n [count=m]]
```

For example:

```
# vxdmpadm -q iostat show dmpnodename=hdisk10

cpu usage = 529us    per cpu memory = 49152b
                        QUEUED I/Os      PENDING I/Os
DMPNODENAME          READS    WRITES
hdisk10              0        0          0
```

To display the count of I/Os that returned with errors on a DMP node, path or controller:

```
# vxdmpadm -e iostat show [filter]
[interval=n [count=m]]
```

For example, to show the I/O counts that returned errors on a path:

```
# vxdmpadm -e iostat show pathname=hdisk55

cpu usage = 656us    per cpu memory = 49152b
                        ERROR I/Os
PATHNAME             READS    WRITES
hdisk55                  0          0
```

Displaying cumulative I/O statistics

Use the `groupby` clause of the `vxdmpadm iostat` command to display cumulative I/O statistics listings per DMP node, controller, array port id, or host-array controller pair and enclosure. If the `groupby` clause is not specified, then the statistics are displayed per path.

To group by DMP node:

```
# vxdmpadm iostat show groupby=dmpnode [all | dmpnodename=dmpnodename
| enclosure=enclr-name]
```

To group by controller:

```
# vxdmpadm iostat show groupby=ctlr [ all | ctlr=ctlr ]
```

For example:

```
# vxdmpadm iostat show groupby=ctlr ctlr=fscsi0

cpu usage = 843us    per cpu memory = 49152b
              OPERATIONS       BLOCKS          AVG TIME(ms)
CTLRNAME   READS  WRITES   READS   WRITES   READS   WRITES
fscsi0     276    0        2205    0         0.03     0.00
```

To group by arrayport:

```
# vxdmpadm iostat show groupby=arrayport [ all | pwwn=array_pwwn
 | enclosure=enclr portid=array-port-id ]
```

For example:

```
# vxdmpadm iostat show groupby=arrayport enclosure=HDS9500-ALUA0 \
portid=1A

            OPERATIONS        BLOCKS       AVG TIME(ms)
PORTNAME   READS    WRITES  READS  WRITES  READS   WRITES
1A         224       14     54      7       4.20    11.10
```

To group by enclosure:

```
# vxdmpadm iostat show groupby=enclosure [ all | enclosure=enclr ]
```

For example:

```
# vxdmpadm iostat show groupby=enclosure enclosure=EMC_CLARiiON0

               OPERATIONS        BLOCKS       AVG TIME(ms)
ENCLRNAME     READS  WRITES  READS  WRITES  READS   WRITES
EMC_CLARiiON 0       0       0      0        0.00     0.00
```

You can also filter out entities for which all data entries are zero. This option is especially useful in a cluster environment which contains many failover devices. You can display only the statistics for the active paths.

To filter all zero entries from the output of the `iostat show` command:

```
# vxdmpadm -z iostat show [all|ctlr=ctlr_name |
dmpnodename=dmp_device_name | enclosure=enclr_name [portid=portid] |
pathname=path_name|pwwn=port_WWN][interval=seconds [count=N]]
```

For example:

```
# vxdmpadm -z iostat show dmpnodename=hdisk40

cpu usage = 906us    per cpu memory = 49152b
             OPERATIONS       BLOCKS          AVG TIME(ms)
PATHNAME  READS  WRITES   READS   WRITES   READS    WRITES
hdisk100  7      0        70      0         0.02      0.00
hdisk115  12     0        58      0         0.03      0.00
hdisk40   10     0        101     0         0.02      0.00
hdisk55   5      0        21      0         0.04      0.00
```

You can now specify the units in which the statistics data is displayed. By default, the read/write times are displayed in milliseconds up to 2 decimal places. The throughput data is displayed in terms of 'BLOCKS' and the output is scaled, meaning that the small values are displayed in small units and the larger values are displayed in bigger units, keeping significant digits constant.The `-u` option accepts the following options:

`k`	Displays throughput in kiloblocks.
`m`	Displays throughput in megablocks.
`g`	Displays throughput in gigablocks.
`bytes`	Displays throughput in exact number of bytes.
`us`	Displays average read/write time in microseconds.

For example: To display average read/write times in microseconds.

```
# vxdmpadm -u us iostat show pathname=hdisk115

cpu usage = 1030us    per cpu memory = 49152b
              OPERATIONS        BLOCKS           AVG TIME(us)
PATHNAME   READS     WRITES  READS     WRITES    READS     WRITES
hdisk115   12        0       58        0         32.00       0.00
```

Setting the attributes of the paths to an enclosure

You can use the `vxdmpadm setattr` command to set the attributes of the paths to an enclosure or disk array.

The attributes set for the paths are persistent and are stored in the file `/etc/vx/dmppolicy.info`.

You can set the following attributes:

`active`

Changes a standby (failover) path to an active path. The following example specifies an active path for an array:

```
# vxdmpadm setattr path hdisk10 pathtype=active
```

`nomanual`

Restores the original primary or secondary attributes of a path. This example restores the path to a JBOD disk:

```
# vxdmpadm setattr path hdisk20 pathtype=nomanual
```

`nopreferred`

Restores the normal priority of a path. The following example restores the default priority to a path:

```
# vxdmpadm setattr path hdisk16 \
  pathtype=nopreferred
```

`preferred [priority=`*N*`]`

Specifies a path as preferred, and optionally assigns a priority number to it. If specified, the priority number must be an integer that is greater than or equal to one. Higher priority numbers indicate that a path is able to carry a greater I/O load.

Note: Setting a priority for path does not change the I/O policy. The I/O policy must be set independently.

See "Specifying the I/O policy" on page 115.

This example first sets the I/O policy to `priority` for an Active/Active disk array, and then specifies a preferred path with an assigned priority of 2:

```
# vxdmpadm setattr enclosure enc0 \
  iopolicy=priority
# vxdmpadm setattr path hdisk16 pathtype=preferred \
  priority=2
```

`primary`	Defines a path as being the primary path for a JBOD disk array. The following example specifies a primary path for a JBOD disk array: `# vxdmpadm setattr path hdisk20 pathtype=primary`

`secondary`	Defines a path as being the secondary path for a JBOD disk array. The following example specifies a secondary path for a JBOD disk array:
	
	`# vxdmpadm setattr path hdisk22 \`
	`pathtype=secondary`

`standby`	Marks a standby (failover) path that it is not used for normal I/O scheduling. This path is used if there are no active paths available for I/O. The next example specifies a standby path for an A/P-C disk array:
	
	`# vxdmpadm setattr path hdisk10 \`
	`pathtype=standby`

Displaying the redundancy level of a device or enclosure

Use the `vxdmpadm getdmpnode` command to list the devices with less than the required redundancy level.

To list the devices on a specified enclosure with fewer than a given number of enabled paths, use the following command:

```
# vxdmpadm getdmpnode enclosure=encl_name redundancy=value
```

For example, to list the devices with fewer than 3 enabled paths, use the following command:

```
# vxdmpadm getdmpnode enclosure=EMC_CLARiiON0 redundancy=3

NAME               STATE    ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
=====================================================================
emc_clariion0_162 ENABLED  EMC_CLARiiON 3     2    1    emc_clariion0
emc_clariion0_182 ENABLED  EMC_CLARiiON 2     2    0    emc_clariion0
emc_clariion0_184 ENABLED  EMC_CLARiiON 3     2    1    emc_clariion0
emc_clariion0_186 ENABLED  EMC_CLARiiON 2     2    0    emc_clariion0
```

To display the minimum redundancy level for a particular device, use the `vxdmpadm getattr` command, as follows:

```
# vxdmpadm getattr enclosure|arrayname|arraytype \
 component-name redundancy
```

For example, to show the minimum redundancy level for the enclosure HDS9500-ALUA0:

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 redundancy

ENCLR_NAME  DEFAULT  CURRENT
=============================================
 HDS9500-ALUA0    0       4
```

Specifying the minimum number of active paths

You can set the minimum redundancy level for a device or an enclosure. The minimum redundancy level is the minimum number of paths that should be active for the device or the enclosure. If the number of paths falls below the minimum redundancy level for the enclosure, a message is sent to the system console and also logged to the DMP log file. Also, notification is sent to `vxnotify` clients.

The value set for minimum redundancy level is stored in the `dmppolicy.info` file, and is persistent. If no minimum redundancy level is set, the default value is 0.

You can use the `vxdmpadm setattr` command to set the minimum redundancy level.

To specify the minimum number of active paths

- Use the `vxdmpadm setattr` command with the redundancy attribute as follows:

  ```
  # vxdmpadm setattr enclosure|arrayname|arraytype component-name
   redundancy=value
  ```

  where *value* is the number of active paths.

  For example, to set the minimum redundancy level for the enclosure HDS9500-ALUA0:

  ```
  # vxdmpadm setattr enclosure HDS9500-ALUA0 redundancy=2
  ```

Displaying the I/O policy

To display the current and default settings of the I/O policy for an enclosure, array or array type, use the `vxdmpadm getattr` command.

The following example displays the default and current setting of `iopolicy` for JBOD disks:

```
# vxdmpadm getattr enclosure Disk iopolicy

ENCLR_NAME        DEFAULT         CURRENT
---------------------------------------
Disk              MinimumQ        Balanced
```

The next example displays the setting of `partitionsize` for the enclosure `enc0`, on which the `balanced` I/O policy with a partition size of 2MB has been set:

```
# vxdmpadm getattr enclosure enc0 partitionsize

ENCLR_NAME      DEFAULT         CURRENT
---------------------------------------
enc0            2048          4096
```

Specifying the I/O policy

You can use the `vxdmpadm setattr` command to change the I/O policy for distributing I/O load across multiple paths to a disk array or enclosure. You can set policies for an enclosure (for example, `HDS01`), for all enclosures of a particular type (such as `HDS`), or for all enclosures of a particular array type (such as `A/A` for Active/Active, or `A/P` for Active/Passive).

Warning: Starting with release 4.1 of VxVM, I/O policies are recorded in the file `/etc/vx/dmppolicy.info`, and are persistent across reboots of the system.

Do not edit this file yourself.

The following policies may be set:

`adaptive`	This policy attempts to maximize overall I/O throughput from/to the disks by dynamically scheduling I/O on the paths. It is suggested for use where I/O loads can vary over time. For example, I/O from/to a database may exhibit both long transfers (table scans) and short transfers (random look ups). The policy is also useful for a SAN environment where different paths may have different number of hops. No further configuration is possible as this policy is automatically managed by DMP.
	
	In this example, the adaptive I/O policy is set for the enclosure `enc1`:
	
	`# vxdmpadm setattr enclosure enc1 \`
	`iopolicy=adaptive`

`balanced [partitionsize=`*`size`*`]`

This policy is designed to optimize the use of caching in disk drives and RAID controllers. The size of the cache typically ranges from 120KB to 500KB or more, depending on the characteristics of the particular hardware. During normal operation, the disks (or LUNs) are logically divided into a number of regions (or partitions), and I/O from/to a given region is sent on only one of the active paths. Should that path fail, the workload is automatically redistributed across the remaining paths.

You can use the size argument to the partitionsize attribute to specify the partition size. The partition size in blocks is adjustable in powers of 2 from 2 up to 231. A value that is not a power of 2 is silently rounded down to the nearest acceptable value.

Specifying a partition size of 0 is equivalent to specifying the default partition size.

The default value for the partition size is 2048 blocks (1024k). Specifying a partition size of 0 is equivalent to the default partition size of 2048 blocks (1024k).

The default value can be changed by adjusting the value of the `dmp_pathswitch_blks_shift` tunable parameter.

See "DMP tunable parameters" on page 170.

Note: The benefit of this policy is lost if the value is set larger than the cache size.

For example, the suggested partition size for an Hitachi HDS 9960 A/A array is from 32,768 to 131,072 blocks (16MB to 64MB) for an I/O activity pattern that consists mostly of sequential reads or writes.

The next example sets the balanced I/O policy with a partition size of 4096 blocks (2MB) on the enclosure enc0:

```
# vxdmpadm setattr enclosure enc0 \
  iopolicy=balanced partitionsize=4096
```

`minimumq`	This policy sends I/O on paths that have the minimum number of outstanding I/O requests in the queue for a LUN. No further configuration is possible as DMP automatically determines the path with the shortest queue.
	
	The following example sets the I/O policy to `minimumq` for a JBOD:
	
	`# vxdmpadm setattr enclosure Disk \`
	`  iopolicy=minimumq`
	
	This is the default I/O policy for all arrays.

`priority`	This policy is useful when the paths in a SAN have unequal performance, and you want to enforce load balancing manually. You can assign priorities to each path based on your knowledge of the configuration and performance characteristics of the available paths, and of other aspects of your system.
	
	See "Setting the attributes of the paths to an enclosure" on page 112.
	
	In this example, the I/O policy is set to `priority` for all SENA arrays:
	
	`# vxdmpadm setattr arrayname SENA \`
	`  iopolicy=priority`

`round-robin`	This policy shares I/O equally between the paths in a round-robin sequence. For example, if there are three paths, the first I/O request would use one path, the second would use a different path, the third would be sent down the remaining path, the fourth would go down the first path, and so on. No further configuration is possible as this policy is automatically managed by DMP.
	
	The next example sets the I/O policy to `round-robin` for all Active/Active arrays:
	
	`# vxdmpadm setattr arraytype A/A \`
	`  iopolicy=round-robin`

`singleactive`	This policy routes I/O down the single active path. This policy can be configured for A/P arrays with one active path per controller, where the other paths are used in case of failover. If configured for A/A arrays, there is no load balancing across the paths, and the alternate paths are only used to provide high availability (HA). If the current active path fails, I/O is switched to an alternate active path. No further configuration is possible as the single active path is selected by DMP.
	
	The following example sets the I/O policy to `singleactive` for JBOD disks:
	
	`# vxdmpadm setattr arrayname Disk \`
	`  iopolicy=singleactive`

Scheduling I/O on the paths of an Asymmetric Active/Active array

You can specify the `use_all_paths` attribute in conjunction with the `adaptive`, `balanced`, `minimumq`, `priority` and `round-robin` I/O policies to specify whether I/O requests are to be scheduled on the secondary paths in addition to the primary paths of an Asymmetric Active/Active (A/A-A) array. Depending on the characteristics of the array, the consequent improved load balancing can increase the total I/O throughput. However, this feature should only be enabled if recommended by the array vendor. It has no effect for array types other than A/A-A.

For example, the following command sets the `balanced` I/O policy with a partition size of 4096 blocks (2MB) on the enclosure `enc0`, and allows scheduling of I/O requests on the secondary paths:

```
# vxdmpadm setattr enclosure enc0 iopolicy=balanced \
      partitionsize=4096 use_all_paths=yes
```

The default setting for this attribute is `use_all_paths=no`.

You can display the current setting for `use_all_paths` for an enclosure, arrayname or arraytype. To do this, specify the `use_all_paths` option to the `vxdmpadm gettattr` command.

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 use_all_paths

ENCLR_NAME  DEFAULT  CURRENT
===========================================
HDS9500-ALUA0 no   yes
```

The `use_all_paths` attribute only applies to A/A-A arrays. For other arrays, the above command displays the message:

```
Attribute is not applicable for this array.
```

Example of applying load balancing in a SAN

This example describes how to configure load balancing in a SAN environment where there are multiple primary paths to an Active/Passive device through several SAN switches. As can be seen in this sample output from the `vxdisk list` command, the device `hdisk18` has eight primary paths:

```
# vxdisk list hdisk18

Device: hdisk18
  .
  .
  .
numpaths: 8
hdisk11 state=enabled type=primary
hdisk12 state=enabled type=primary
hdisk13 state=enabled type=primary
hdisk14 state=enabled type=primary
hdisk15 state=enabled type=primary
hdisk16 state=enabled type=primary
hdisk17 state=enabled type=primary
hdisk18 state=enabled type=primary
```

In addition, the device is in the enclosure `ENC0`, belongs to the disk group `mydg`, and contains a simple concatenated volume `myvol1`.

The first step is to enable the gathering of DMP statistics:

```
# vxdmpadm iostat start
```

Next the `dd` command is used to apply an input workload from the volume:

```
# dd if=/dev/vx/rdsk/mydg/myvol1 of=/dev/null &
```

By running the `vxdmpadm iostat` command to display the DMP statistics for the device, it can be seen that all I/O is being directed to one path, `hdisk18`:

```
# vxdmpadm iostat show dmpnodename=hdisk18 interval=5 count=2
    .
    .
    .
```

```
cpu usage = 11294us per cpu memory = 32768b
               OPERATIONS          KBYTES         AVG TIME(ms)
PATHNAME    READS    WRITES    READS    WRITES    READS       WRITES
hdisk11     0        0         0        0          0.00         0.00
hdisk12     0        0         0        0          0.00         0.00
hdisk13     0        0         0        0          0.00         0.00
hdisk14     0        0         0        0          0.00         0.00
hdisk15     0        0         0        0          0.00         0.00
hdisk16     0        0         0        0          0.00         0.00
hdisk17     0        0         0        0          0.00         0.00
hdisk18     10986    0         5493     0          0.41         0.00
```

The `vxdmpadm` command is used to display the I/O policy for the enclosure that contains the device:

```
# vxdmpadm getattr enclosure ENC0 iopolicy

ENCLR_NAME     DEFAULT          CURRENT
============================================
ENC0           MinimumQ      Single-Active
```

This shows that the policy for the enclosure is set to `singleactive`, which explains why all the I/O is taking place on one path.

To balance the I/O load across the multiple primary paths, the policy is set to `round-robin` as shown here:

```
# vxdmpadm setattr enclosure ENC0 iopolicy=round-robin
# vxdmpadm getattr enclosure ENC0 iopolicy

ENCLR_NAME    DEFAULT            CURRENT
============================================
ENC0          MinimumQ           Round-Robin
```

The DMP statistics are now reset:

```
# vxdmpadm iostat reset
```

With the workload still running, the effect of changing the I/O policy to balance the load across the primary paths can now be seen.

```
# vxdmpadm iostat show dmpnodename=hdisk18 interval=5 count=2
    .
    .
    .
cpu usage = 14403us per cpu memory = 32768b
```

```
                OPERATIONS          KBYTES          AVG TIME(ms)
PATHNAME     READS   WRITES    READS    WRITES    READS      WRITES
hdisk11      2041    0         1021     0          0.39        0.00
hdisk12      1894    0         947      0          0.39        0.00
hdisk13      2008    0         1004     0          0.39        0.00
hdisk14      2054    0         1027     0          0.40        0.00
hdisk15      2171    0         1086     0          0.39        0.00
hdisk16      2095    0         1048     0          0.39        0.00
hdisk17      2073    0         1036     0          0.39        0.00
hdisk18      2042    0         1021     0          0.39        0.00
```

The enclosure can be returned to the single active I/O policy by entering the following command:

`# vxdmpadm setattr enclosure ENC0 iopolicy=singleactive`

Disabling I/O for paths, controllers or array ports

Disabling I/O through a path, HBA controller or array port prevents DMP from issuing I/O requests through the specified path, or the paths that are connected to the specified controller or array port. The command blocks until all pending I/O requests issued through the paths are completed.

To disable I/O for a path, use the following command:

`# vxdmpadm [-c|-f] disable path=path_name`

To disable I/O for multiple paths, use the following command:

`# vxdmpadm [-c|-f] disable path=path_name1,path_name2,path_nameN`

To disable I/O for the paths connected to an HBA controller, use the following command:

`# vxdmpadm [-c|-f] disable ctlr=ctlr_name`

To disable I/O for the paths connected to an array port, use one of the following commands:

```
# vxdmpadm [-c|-f] disable enclosure=enclr_name portid=array_port_ID
# vxdmpadm [-c|-f] disable pwwn=array_port_WWN
```

where the array port is specified either by the enclosure name and the array port ID, or by the array port's worldwide name (WWN) identifier.

The following are examples of using the command to disable I/O on an array port:

```
# vxdmpadm disable enclosure=HDS9500V0 portid=1A
# vxdmpadm disable pwwn=20:00:00:E0:8B:06:5F:19
```

You can use the -c option to check if there is only a single active path to the disk. If so, the `disable` command fails with an error message unless you use the `-f` option to forcibly disable the path.

The `disable` operation fails if it is issued to a controller that is connected to the root disk through a single path, and there are no root disk mirrors configured on alternate paths. If such mirrors exist, the command succeeds.

Enabling I/O for paths, controllers or array ports

Enabling a controller allows a previously disabled path, HBA controller or array port to accept I/O again. This operation succeeds only if the path, controller or array port is accessible to the host, and I/O can be performed on it. When connecting Active/Passive disk arrays, the `enable` operation results in failback of I/O to the primary path. The `enable` operation can also be used to allow I/O to the controllers on a system board that was previously detached.

Note: From release 5.0 of VxVM, this operation is supported for controllers that are used to access disk arrays on which cluster-shareable disk groups are configured.

To enable I/O for a path, use the following command:

```
# vxdmpadm enable path=path_name
```

To enable I/O for multiple paths, use the following command:

```
# vxdmpadm enable path=path_name1,path_name2,path_nameN
```

To enable I/O for the paths connected to an HBA controller, use the following command:

```
# vxdmpadm enable ctlr=ctlr_name
```

To enable I/O for the paths connected to an array port, use one of the following commands:

```
# vxdmpadm enable enclosure=enclr_name portid=array_port_ID
# vxdmpadm [-f] enable pwwn=array_port_WWN
```

where the array port is specified either by the enclosure name and the array port ID, or by the array port's worldwide name (WWN) identifier.

The following are examples of using the command to enable I/O on an array port:

```
# vxdmpadm enable enclosure=HDS9500V0 portid=1A
# vxdmpadm enable pwwn=20:00:00:E0:8B:06:5F:19
```

Renaming an enclosure

The `vxdmpadm setattr` command can be used to assign a meaningful name to an existing enclosure, for example:

```
# vxdmpadm setattr enclosure enc0 name=GRP1
```

This example changes the name of an enclosure from `enc0` to `GRP1`.

Note: The maximum length of the enclosure name prefix is 23 characters.

The following command shows the changed name:

```
# vxdmpadm listenclosure all

ENCLR_NAME     ENCLR_TYPE     ENCLR_SNO                 STATUS
============================================================
other0         OTHER         OTHER_DISKS                CONNECTED
jbod0          X1            X1_DISKS                   CONNECTED
GRP1           ACME          60020f20000001a90000       CONNECTED
```

Configuring the response to I/O failures

You can configure how DMP responds to failed I/O requests on the paths to a specified enclosure, disk array name, or type of array. By default, DMP is configured to retry a failed I/O request up to five times for a single path.

To display the current settings for handling I/O request failures that are applied to the paths to an enclosure, array name or array type, use the `vxdmpadm getattr` command.

See "Displaying recovery option values" on page 127.

To set a limit for the number of times that DMP attempts to retry sending an I/O request on a path, use the following command:

```
# vxdmpadm setattr \
  {enclosure enc-name|arrayname name|arraytype type} \
  recoveryoption=fixedretry retrycount=n
```

The value of the argument to `retrycount` specifies the number of retries to be attempted before DMP reschedules the I/O request on another available path, or fails the request altogether.

As an alternative to specifying a fixed number of retries, you can specify the amount of time DMP allows for handling an I/O request. If the I/O request does not succeed within that time, DMP fails the I/O request. To specify an `iotimeout` value, use the following command:

```
# vxdmpadm setattr \
  {enclosure enc-name|arrayname name|arraytype type} \
  recoveryoption=timebound iotimeout=seconds
```

The default value of `iotimeout` is 300 seconds. For some applications such as Oracle, it may be desirable to set `iotimeout` to a larger value. The `iotimeout` value for DMP should be greater than the I/O service time of the underlying operating system layers.

Note: The `fixedretry` and `timebound` settings are mutually exclusive.

The following example configures time-bound recovery for the enclosure `enc0`, and sets the value of `iotimeout` to 360 seconds:

```
# vxdmpadm setattr enclosure enc0 recoveryoption=timebound \
  iotimeout=360
```

The next example sets a fixed-retry limit of 10 for the paths to all Active/Active arrays:

```
# vxdmpadm setattr arraytype A/A recoveryoption=fixedretry \
  retrycount=10
```

Specifying `recoveryoption=default` resets DMP to the default settings corresponding to `recoveryoption=fixedretry retrycount=5`, for example:

```
# vxdmpadm setattr arraytype A/A recoveryoption=default
```

The above command also has the effect of configuring I/O throttling with the default settings.

See "Configuring the I/O throttling mechanism" on page 126.

Note: The response to I/O failure settings is persistent across reboots of the system.

Configuring the I/O throttling mechanism

By default, DMP is configured with I/O throttling turned off for all paths. To display the current settings for I/O throttling that are applied to the paths to an enclosure, array name or array type, use the `vxdmpadm getattr` command.

See "Displaying recovery option values" on page 127.

If enabled, I/O throttling imposes a small overhead on CPU and memory usage because of the activity of the statistics-gathering daemon. If I/O throttling is disabled, the daemon no longer collects statistics, and remains inactive until I/O throttling is re-enabled.

To turn off I/O throttling, use the following form of the `vxdmpadm setattr` command:

```
# vxdmpadm setattr \
  {enclosure enc-name|arrayname name|arraytype type} \
  recoveryoption=nothrottle
```

The following example shows how to disable I/O throttling for the paths to the enclosure `enc0`:

```
# vxdmpadm setattr enclosure enc0 recoveryoption=nothrottle
```

The `vxdmpadm setattr` command can be used to enable I/O throttling on the paths to a specified enclosure, disk array name, or type of array:

```
# vxdmpadm setattr \
  {enclosure enc-name|arrayname name|arraytype type}\
  recoveryoption=throttle [iotimeout=seconds]
```

If the `iotimeout` attribute is specified, its argument specifies the time in seconds that DMP waits for an outstanding I/O request to succeed before invoking I/O throttling on the path. The default value of `iotimeout` is 10 seconds. Setting `iotimeout` to a larger value potentially causes more I/O requests to become queued up in the SCSI driver before I/O throttling is invoked.

The following example sets the value of `iotimeout` to 60 seconds for the enclosure `enc0`:

```
# vxdmpadm setattr enclosure enc0 recoveryoption=throttle \
  iotimeout=60
```

Specify `recoveryoption=default` to reset I/O throttling to the default settings, as follows:

```
# vxdmpadm setattr arraytype A/A recoveryoption=default
```

The above command configures the default behavior, corresponding to `recoveryoption=nothrottle`. The above command also configures the default behavior for the response to I/O failures.

See "Configuring the response to I/O failures" on page 124.

Note: The I/O throttling settings are persistent across reboots of the system.

Configuring Subpaths Failover Groups (SFG)

The Subpaths Failover Groups (SFG) feature can be turned on or off using the tunable `dmp_sfg_threshold`.

To turn off the feature, set the tunable `dmp_sfg_threshold` value to 0:

```
# vxdmpadm settune dmp_sfg_threshold=0
```

To turn on the feature, set the `dmp_sfg_threshold` value to the required number of path failures which triggers SFG. The default is 1.

```
# vxdmpadm settune dmp_sfg_threshold=N
```

The default value of the tunable is "1" which represents that the feature is on.

To see the Subpaths Failover Groups ID, use the following command:

```
# vxdmpadm getportids {ctlr=ctlr_name | dmpnodename=dmp_device_name \
   | enclosure=enclr_name | path=path_name}
```

Configuring Low Impact Path Probing

The Low Impact Path Probing (LIPP) feature can be turned on or off using the `vxdmpadm settune` command:

```
# vxdmpadm settune dmp_low_impact_probe=[on|off]
```

Path probing will be optimized by probing a subset of paths connected to same HBA and array port. The size of the subset of paths can be controlled by the `dmp_probe_threshold` tunable. The default value is set to 5.

```
# vxdmpadm settune dmp_probe_threshold=N
```

Displaying recovery option values

To display the current settings for handling I/O request failures that are applied to the paths to an enclosure, array name or array type, use the following command:

```
# vxdmpadm getattr \
  {enclosure enc-name|arrayname name|arraytype type} \
  recoveryoption
```

The following example shows the `vxdmpadm getattr` command being used to display the `recoveryoption` option values that are set on an enclosure.

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 recoveryoption
ENCLR-NAME      RECOVERY-OPTION   DEFAULT[VAL]    CURRENT[VAL]
===============================================================
HDS9500-ALUA0   Throttle          Nothrottle[0]   Timebound[60]
HDS9500-ALUA0   Error-Retry       Fixed-Retry[5]  Timebound[20]
```

This shows the default and current policy options and their values.

Table 4-1 summarizes the possible recovery option settings for retrying I/O after an error.

Table 4-1 Recovery options for retrying I/O after an error

Recovery option	Possible settings	Description
recoveryoption=fixedretry	Fixed-Retry (retrycount)	DMP retries a failed I/O request for the specified number of times if I/O fails.
recoveryoption=timebound	Timebound (iotimeout)	DMP retries a failed I/O request for the specified time in seconds if I/O fails.

Table 4-2 summarizes the possible recovery option settings for throttling I/O.

Table 4-2 Recovery options for I/O throttling

Recovery option	Possible settings	Description
recoveryoption=nothrottle	None	I/O throttling is not used.
recoveryoption=throttle	Timebound (iotimeout)	DMP throttles the path if an I/O request does not return within the specified time in seconds.

Configuring DMP path restoration policies

DMP maintains a kernel thread that re-examines the condition of paths at a specified interval. The type of analysis that is performed on the paths depends on the checking policy that is configured.

Note: The DMP path restoration thread does not change the disabled state of the path through a controller that you have disabled using `vxdmpadm disable`.

When configuring DMP path restoration policies, you must stop the path restoration thread, and then restart it with new attributes.

See "Stopping the DMP path restoration thread" on page 130.

Use the `vxdmpadm start restore` command to configure one of the following restore policies. The policy will remain in effect until the restore thread is stopped or the values are changed using `vxdmpadm settune` command.

- `check_all`

  The path restoration thread analyzes all paths in the system and revives the paths that are back online, as well as disabling the paths that are inaccessible. The command to configure this policy is:

  ```
  # vxdmpadm start restore [interval=seconds] policy=check_all
  ```

- `check_alternate`

  The path restoration thread checks that at least one alternate path is healthy. It generates a notification if this condition is not met. This policy avoids inquiry commands on all healthy paths, and is less costly than `check_all` in cases where a large number of paths are available. This policy is the same as `check_all` if there are only two paths per DMP node. The command to configure this policy is:

  ```
  # vxdmpadm start restore [interval=seconds] \
    policy=check_alternate
  ```

- `check_disabled`

  This is the default path restoration policy. The path restoration thread checks the condition of paths that were previously disabled due to hardware failures, and revives them if they are back online. The command to configure this policy is:

  ```
  # vxdmpadm start restore [interval=seconds] \
    policy=check_disabled
  ```

- `check_periodic`

  The path restoration thread performs `check_all` once in a given number of cycles, and `check_disabled` in the remainder of the cycles. This policy may lead to periodic slowing down (due to `check_all`) if there is a large number of paths available. The command to configure this policy is:

  ```
  # vxdmpadm start restore interval=seconds \
    policy=check_periodic [period=number]
  ```

  The `interval` attribute must be specified for this policy. The default number of cycles between running the `check_all` policy is 10.

The interval attribute specifies how often the path restoration thread examines the paths. For example, after stopping the path restoration thread, the polling interval can be set to 400 seconds using the following command:

```
# vxdmpadm start restore interval=400
```

Starting with the 5.0MP3 release, you can also use the `vxdmpadm settune` command to change the restore policy, restore interval, and restore period. This method stores the values for these arguments as DMP tunables. The settings are immediately applied and are persistent across reboots. Use the `vxdmpadm gettune` to view the current settings.

See "DMP tunable parameters" on page 170.

If the `vxdmpadm start restore` command is given without specifying a policy or interval, the path restoration thread is started with the persistent policy and interval settings previously set by the administrator with the `vxdmpadm settune` command. If the administrator has not set a policy or interval, the system defaults are used. The system default restore policy is `check_disabled`. The system default interval is 300 seconds.

Warning: Decreasing the interval below the system default can adversely affect system performance.

Stopping the DMP path restoration thread

Use the following command to stop the DMP path restoration thread:

```
# vxdmpadm stop restore
```

Warning: Automatic path failback stops if the path restoration thread is stopped.

Displaying the status of the DMP path restoration thread

Use the following command to display the status of the automatic path restoration kernel thread, its polling interval, and the policy that it uses to check the condition of paths:

```
# vxdmpadm stat restored
```

This produces output such as the following:

```
The number of daemons running : 1
The interval of daemon: 300
The policy of daemon: check_disabled
```

Displaying information about the DMP error-handling thread

To display information about the kernel thread that handles DMP errors, use the following command:

```
# vxdmpadm stat errord
```

One daemon should be shown as running.

Configuring array policy modules

An array policy module (APM) is a dynamically loadable kernel module (plug-in for DMP) for use in conjunction with an array. An APM defines array-specific procedures and commands to:

- Select an I/O path when multiple paths to a disk within the array are available.
- Select the path failover mechanism.
- Select the alternate path in the case of a path failure.
- Put a path change into effect.
- Respond to SCSI reservation or release requests.

DMP supplies default procedures for these functions when an array is registered. An APM may modify some or all of the existing procedures that are provided by DMP or by another version of the APM.

You can use the following command to display all the APMs that are configured for a system:

```
# vxdmpadm listapm all
```

The output from this command includes the file name of each module, the supported array type, the APM name, the APM version, and whether the module is currently loaded and in use. To see detailed information for an individual module, specify the module name as the argument to the command:

```
# vxdmpadm listapm module_name
```

To add and configure an APM, use the following command:

```
# vxdmpadm -a cfgapm module_name [attr1=value1 \
      [attr2=value2 ...]]
```

The optional configuration attributes and their values are specific to the APM for an array. Consult the documentation that is provided by the array vendor for details.

Note: By default, DMP uses the most recent APM that is available. Specify the `-u` option instead of the `-a` option if you want to force DMP to use an earlier version of the APM. The current version of an APM is replaced only if it is not in use.

Specifying the `-r` option allows you to remove an APM that is not currently loaded:

```
# vxdmpadm -r cfgapm module_name
```

See the `vxdmpadm`(1M) manual page.

Chapter 5

Administering disks

This chapter includes the following topics:

About disk management

Veritas Dynamic Multi-Pathing (DMP) is used to administer multiported disk arrays.

See "How DMP works" on page 14.

DMP uses the Device Discovery Layer (DDL) to handle device discovery and configuration of disk arrays. DDL discovers disks and their attributes that are required for DMP operations. Use the `vxddladm` utility to administer the DDL.

See "How to administer the Device Discovery Layer" on page 139.

Discovering and configuring newly added disk devices

When you physically connect new disks to a host or when you zone new fibre channel devices to a host, you can use the `vxdctl enable` command to rebuild the volume device node directories and to update the DMP internal database to reflect the new state of the system.

To reconfigure the DMP database, first run `cfgmgr` to make the operating system recognize the new disks, and then invoke the `vxdctl enable` command.

You can also use the `vxdisk scandisks` command to scan devices in the operating system device tree, and to initiate dynamic reconfiguration of multipathed disks.

If you want DMP to scan only for new devices that have been added to the system, and not for devices that have been enabled or disabled, specify the `-f` option to either of the commands, as shown here:

```
# vxdctl -f enable
# vxdisk -f scandisks
```

However, a complete scan is initiated if the system configuration has been modified by changes to:

- Installed array support libraries.
- The list of devices that are excluded from use by VxVM.
- DISKS (JBOD), SCSI3, or foreign device definitions.

See the `vxdctl`(1M) manual page.

See the `vxdisk`(1M) manual page.

Partial device discovery

Dynamic Multi-Pathing (DMP) supports partial device discovery where you can include or exclude sets of disks or disks attached to controllers from the discovery process.

The `vxdisk scandisks` command rescans the devices in the OS device tree and triggers a DMP reconfiguration. You can specify parameters to `vxdisk scandisks` to implement partial device discovery. For example, this command makes DMP discover newly added devices that were unknown to it earlier:

```
# vxdisk scandisks new
```

The next example discovers fabric devices:

```
# vxdisk scandisks fabric
```

The following command scans for the devices `hdisk10` and `hdisk11`:

```
# vxdisk scandisks device=hdisk10,hdisk11
```

Alternatively, you can specify a ! prefix character to indicate that you want to scan for all devices except those that are listed.

Note: The ! character is a special character in some shells. The following examples show how to escape it in a bash shell.

```
# vxdisk scandisks \!device=hdisk10,hdisk11
```

You can also scan for devices that are connected (or not connected) to a list of logical or physical controllers. For example, this command discovers and configures all devices except those that are connected to the specified logical controllers:

```
# vxdisk scandisks \!ctlr=scsi1,scsi2
```

The next command discovers devices that are connected to the specified physical controller:

```
# vxdisk scandisks pctlr=10-60
```

The items in a list of physical controllers are separated by + characters.

You can use the command `vxdmpadm getctlr all` to obtain a list of physical controllers.

You should specify only one selection argument to the `vxdisk scandisks` command. Specifying multiple options results in an error.

See the `vxdisk`(1M) manual page.

Discovering disks and dynamically adding disk arrays

DMP uses array support libraries (ASLs) to provide array-specific support for multi-pathing. An array support library (ASL) is a dynamically loadable shared library (plug-in for DDL). The ASL implements hardware-specific logic to discover device attributes during device discovery. DMP provides the device discovery layer (DDL) to determine which ASLs should be associated to each disk array

In some cases, DMP can also provide basic multi-pathing and failover functionality by treating LUNs as disks (JBODs).

How DMP claims devices

For fully optimized support of any array and for support of more complicated array types, DMP requires the use of array-specific array support libraries (ASLs), possibly coupled with array policy modules (APMs). ASLs and APMs effectively are array-specific plugins that allow close tie-in of DMP with any specific array model.

See the Hardware Compatibility List for the complete list of supported arrays.

http://entsupport.symantec.com/docs/330441

During device discovery, the DDL checks the installed ASL for each device to find which ASL claims the device. If no ASL is found to claim the device, the DDL checks for a corresponding JBOD definition. You can add JBOD definitions for unsupported arrays to enable DMP to provide multi-pathing for the array. If a JBOD definition is found, the DDL claims the devices in the DISKS category, which adds the LUNs to the list of JBOD (physical disk) devices used by DMP. If the JBOD definition includes a cabinet number, DDL uses the cabinet number to group the LUNs into enclosures.

See "Adding unsupported disk arrays to the DISKS category" on page 146.

DMP can provide basic multi-pathing to ALUA-compliant arrays even if there is no ASL or JBOD definition. DDL claims the LUNs as part of the aluadisk enclosure. The array type is shown as ALUA. Adding a JBOD definition also enables you to group the LUNs into enclosures.

Disk categories

Disk arrays that have been certified for use with Veritas Volume Manager are supported by an array support library (ASL), and are categorized by the vendor ID string that is returned by the disks (for example, "`HITACHI`").

Disks in JBODs which are capable of being multipathed by DMP, are placed in the `DISKS` category. Disks in unsupported arrays can also be placed in the `DISKS` category.

See "Adding unsupported disk arrays to the DISKS category" on page 146.

Disks in JBODs that do not fall into any supported category, and which are not capable of being multipathed by DMP are placed in the `OTHER_DISKS` category.

Adding support for a new disk array

You can dynamically add support for a new type of disk array. The support comes in the form of Array Support Libraries (ASLs) that are developed by Symantec. Symantec provides support for new disk arrays through updates to the `VRTSaslapm` fileset. To determine if an updated `VRTSaslapm` fileset is available for download, refer to the hardware compatibility list tech note. The hardware compatibility list provides a link to the latest fileset for download and instructions for installing the `VRTSaslapm` fileset. You can upgrade the `VRTSaslapm` fileset while the system is online; you do not need to stop the applications.

To access the hardware compatibility list, go to the following URL:

http://entsupport.symantec.com/docs/330441

The new disk array does not need to be already connected to the system when the `VRTSaslapm` fileset is installed. If any of the disks in the new disk array are subsequently connected, you need to trigger OS device discovery using the `cfgmgr` command and then trigger DDL device discovery using the `vxdctl enable` command.

See "Adding new LUNs dynamically to a new target ID" on page 158.

If you need to remove the latest `VRTSaslapm` fileset, you can revert to the previously installed version. For the detailed procedure, refer to the *Veritas Volume Manager Troubleshooting Guide*.

Enabling discovery of new disk arrays

The `vxdctl enable` command scans all of the disk devices and their attributes, updates the DMP device list, and reconfigures DMP with the new device database. There is no need to reboot the host.

Warning: This command ensures that Dynamic Multi-Pathing is set up correctly for the array. Otherwise, VxVM treats the independent paths to the disks as separate devices, which can result in data corruption.

To enable discovery of a new disk array

◆ Type the following command:

```
# vxdctl enable
```

Third-party driver coexistence

The third-party driver (TPD) coexistence feature of DMP allows I/O that is controlled by some third-party multi-pathing drivers to bypass DMP while retaining the monitoring capabilities of DMP. If a suitable ASL is available and installed, devices that use TPDs can be discovered without requiring you to set up a specification file, or to run a special command. In previous releases, VxVM only supported TPD coexistence if the code of the third-party driver was intrusively modified. Now, the TPD coexistence feature maintains backward compatibility with such methods, but it also permits coexistence without requiring any change in a third-party multi-pathing driver.

See "Displaying information about TPD-controlled devices" on page 102.

Autodiscovery of EMC Symmetrix arrays

In VxVM 4.0, there were two possible ways to configure EMC Symmetrix arrays:

- With EMC PowerPath installed, EMC Symmetrix arrays could be configured as foreign devices.
  See "Foreign devices" on page 149.
- Without EMC PowerPath installed, DMP could be used to perform multi-pathing.

On upgrading a system to VxVM 4.1 or later release, existing EMC PowerPath devices can be discovered by DDL, and configured into DMP as autoconfigured disks with DMP nodes, even if PowerPath is being used to perform multi-pathing. There is no need to configure such arrays as foreign devices.

Table 5-1 shows the scenarios for using DMP with PowerPath.

The ASLs are all included in the ASL-APM fileset, which is installed when you install Storage Foundation products.

Table 5-1 Scenarios for using DMP with PowerPath

PowerPath	DMP	Array configuration mode
Installed.	The `libvxpp` ASL handles EMC Symmetrix arrays and DGC CLARiiON claiming internally. PowerPath handles failover.	EMC Symmetrix - Any DGC CLARiiON - Active/Passive (A/P), Active/Passive in Explicit Failover mode (A/P-F) and ALUA Explicit failover
Not installed; the array is EMC Symmetrix.	DMP handles multi-pathing. The ASL name is `libvxemc`.	Active/Active
Not installed; the array is DGC CLARiioN (CXn00).	DMP handles multi-pathing. The ASL name is `libvxCLARiiON`.	Active/Passive (A/P), Active/Passive in Explicit Failover mode (A/P-F) and ALUA

If any EMCpower disks are configured as foreign disks, use the `vxddladm rmforeign` command to remove the foreign definitions, as shown in this example:

```
# vxddladm rmforeign blockpath=/dev/emcpower10 \
  charpath=/dev/emcpower10
```

To allow DMP to receive correct inquiry data, the Common Serial Number (C-bit) Symmetrix Director parameter must be set to enabled.

How to administer the Device Discovery Layer

The Device Discovery Layer (DDL) allows dynamic addition of disk arrays. DDL discovers disks and their attributes that are required for DMP operations.

The DDL is administered using the `vxddladm` utility to perform the following tasks:

- List the hierarchy of all the devices discovered by DDL including iSCSI devices.
- List all the Host Bus Adapters including iSCSI
- List the ports configured on a Host Bus Adapter
- List the targets configured from a Host Bus Adapter
- List the devices configured from a Host Bus Adapter
- Get or set the iSCSI operational parameters
- List the types of arrays that are supported.
- Add support for an array to DDL.
- Remove support for an array from DDL.
- List information about excluded disk arrays.
- List disks that are supported in the `DISKS` (JBOD) category.
- Add disks from different vendors to the `DISKS` category.
- Remove disks from the `DISKS` category.
- Add disks as foreign devices.

The following sections explain these tasks in more detail.

See the `vxddladm`(1M) manual page.

Listing all the devices including iSCSI

You can display the hierarchy of all the devices discovered by DDL, including iSCSI devices.

To list all the devices including iSCSI

- Type the following command:

  `# vxddladm list`

  The following is a sample output:

```
HBA fscsi0 (20:00:00:E0:8B:19:77:BE)
     Port fscsi0_p0 (50:0A:09:80:85:84:9D:84)
          Target fscsi0_p0_t0 (50:0A:09:81:85:84:9D:84)
               LUN hdisk1
. . .
HBA iscsi0 (iqn.1986-03.com.sun:01:0003ba8ed1b5.45220f80)
     Port  iscsi0_p0 (10.216.130.10:3260)
          Target  iscsi0_p0_t0 (iqn.1992-08.com.netapp:sn.84188548)
               LUN hdisk2
               LUN hdisk3
          Target  iscsi0_p0_t1 (iqn.1992-08.com.netapp:sn.84190939)
. . .
```

Listing all the Host Bus Adapters including iSCSI

You can obtain information about all the Host Bus Adapters configured on the system, including iSCSI adapters. This includes the following information:

Driver	Driver controlling the HBA.
Firmware	Firmware version.
Discovery	The discovery method employed for the targets.
State	Whether the device is Online or Offline.
Address	The hardware address.

To list all the Host Bus Adapters including iSCSI

- Use the following command to list all of the HBAs, including iSCSI devices, configured on the system:

  `# vxddladm list hbas`

Listing the ports configured on a Host Bus Adapter

You can obtain information about all the ports configured on an HBA. The display includes the following information:

HBA-ID	The parent HBA.
State	Whether the device is Online or Offline.
Address	The hardware address.

To list the ports configured on a Host Bus Adapter

◆ Use the following command to obtain the ports configured on an HBA:

```
# vxddladm list ports

PortID   HBA-ID    State      Address
-------------------------------------------------------------------
fscsi0_p0    fscsi0        Online     50:0A:09:80:85:84:9D:84
iscsi0_p0    iscsi0        Online     10.216.130.10:3260
```

Listing the targets configured from a Host Bus Adapter or a port

You can obtain information about all the targets configured from a Host Bus Adapter or a port. This includes the following information:

Alias	The alias name, if available.
HBA-ID	Parent HBA or port.
State	Whether the device is Online or Offline.
Address	The hardware address.

To list the targets

◆ To list all of the targets, use the following command:

```
# vxddladm list targets
```

The following is a sample output:

```
TgtID          Alias  HBA-ID   State   Address
-----------------------------------------------------------------
fscsi0_p0_t0   -      fscsi0   Online  50:0A:09:80:85:84:9D:84
iscsi0_p0_t1   -      iscsi0   Online  iqn.1992-08.com.netapp:sn.84190939
```

To list the targets configured from a Host Bus Adapter or port

- You can filter based on a HBA or port, using the following command:

```
# vxddladm list targets [hba=hba_name|port=port_name]
```

  For example, to obtain the targets configured from the specified HBA:

```
# vxddladm list targets hba=fscsi0

TgtID          Alias  HBA-ID   State   Address
-----------------------------------------------------------------
fscsi0_p0_t0   -      fscsi0   Online  50:0A:09:80:85:84:9D:84
```

Listing the devices configured from a Host Bus Adapter and target

You can obtain information about all the devices configured from a Host Bus Adapter. This includes the following information:

Target-ID	The parent target.
State	Whether the device is Online or Offline.
DDL status	Whether the device is claimed by DDL. If claimed, the output also displays the ASL name.

To list the devices configured from a Host Bus Adapter

- To obtain the devices configured, use the following command:

```
# vxddladm list devices

Device    Target-ID   State    DDL status (ASL)
------------------------------------------------------------
hdisk1    fscsi0_p0_t0  Online   CLAIMED (libvxemc.so)
hdisk2    fscsi0_p0_t0  Online   SKIPPED
hdisk3    fscsi0_p0_t0  Offline  ERROR
hdisk4    fscsi0_p0_t0  Online   EXCLUDED
hdisk5    fscsi0_p0_t0  Offline  MASKED
```

To list the devices configured from a Host Bus Adapter and target

- To obtain the devices configured from a particular HBA and target, use the following command:

```
# vxddladm list devices target=target_name
```

Getting or setting the iSCSI operational parameters

DDL provides an interface to set and display certain parameters that affect the performance of the iSCSI device path. However, the underlying OS framework must support the ability to set these values. The `vxddladm set` command returns an error if the OS support is not available.

Table 5-2 Parameters for iSCSI devices

Parameter	Default value	Minimum value	Maximum value
DataPDUInOrder	yes	no	yes
DataSequenceInOrder	yes	no	yes
DefaultTime2Retain	20	0	3600
DefaultTime2Wait	2	0	3600
ErrorRecoveryLevel	0	0	2
FirstBurstLength	65535	512	16777215
InitialR2T	yes	no	yes
ImmediateData	yes	no	yes
MaxBurstLength	262144	512	16777215
MaxConnections	1	1	65535
MaxOutStandingR2T	1	1	65535
MaxRecvDataSegmentLength	8182	512	16777215

To get the iSCSI operational parameters on the initiator for a specific iSCSI target

◆ Type the following commands:

```
# vxddladm getiscsi target=tgt-id {all | parameter}
```

You can use this command to obtain all the iSCSI operational parameters. The following is a sample output:

```
# vxddladm getiscsi target=iscsi0_p2_t0

PARAMETER                 CURRENT      DEFAULT       MIN          MAX
------------------------------------------------------------------------------
DataPDUInOrder            yes            yes          no            yes
DataSequenceInOrder       yes            yes          no            yes
DefaultTime2Retain        20              20           0           3600
DefaultTime2Wait           2               2           0           3600
ErrorRecoveryLevel         0               0           0              2
FirstBurstLength          65535        65535         512       16777215
InitialR2T                yes            yes          no            yes
ImmediateData             yes            yes          no            yes
MaxBurstLength            262144      262144         512       16777215
MaxConnections            1                1           1          65535
MaxOutStandingR2T         1                1           1          65535
MaxRecvDataSegmentLength  8192          8182         512       16777215
```

To set the iSCSI operational parameters on the initiator for a specific iSCSI target

◆ Type the following command:

```
# vxddladm setiscsi target=tgt-id
   parameter=value
```

Listing all supported disk arrays

Use this procedure to obtain values for the `vid` and `pid` attributes that are used with other forms of the `vxddladm` command.

To list all supported disk arrays

◆ Type the following command:

```
# vxddladm listsupport all
```

Excluding support for a disk array library

To exclude support for a disk array library

- Type the following command:

  ```
  # vxddladm excludearray libname=libvxenc.so
  ```

  This example excludes support for disk arrays that depends on the library `libvxenc.so`. You can also exclude support for disk arrays from a particular vendor, as shown in this example:

  ```
  # vxddladm excludearray vid=ACME pid=X1
  ```

Re-including support for an excluded disk array library

To re-include support for an excluded disk array library

- If you have excluded support for all arrays that depend on a particular disk array library, you can use the `includearray` keyword to remove the entry from the exclude list, as shown in the following example:

  ```
  # vxddladm includearray libname=libvxenc.so
  ```

Listing excluded disk arrays

To list all disk arrays that are currently excluded from use by VxVM

- Type the following command:

  ```
  # vxddladm listexclude
  ```

Listing supported disks in the DISKS category

To list disks that are supported in the `DISKS` (JBOD) category

- Type the following command:

  ```
  # vxddladm listjbod
  ```

Displaying details about a supported array library

To display details about a supported array library

- Type the following command:

  ```
  # vxddladm listsupport libname=library_name.so
  ```

Adding unsupported disk arrays to the DISKS category

Disk arrays should be added as JBOD devices if no ASL is available for the array.

JBODs are assumed to be Active/Active (A/A) unless otherwise specified. If a suitable ASL is not available, an A/A-A, A/P or A/PF array must be claimed as an Active/Passive (A/P) JBOD to prevent path delays and I/O failures. If a JBOD is ALUA-compliant, it is added as an ALUA array.

See "How DMP works" on page 14.

Warning: This procedure ensures that Dynamic Multi-Pathing (DMP) is set up correctly on an array that is not supported by Veritas Volume Manager. Otherwise, Veritas Volume Manager treats the independent paths to the disks as separate devices, which can result in data corruption.

To add an unsupported disk array to the DISKS category

1. Use the following command to identify the vendor ID and product ID of the disks in the array:

   ```
   # /etc/vx/diag.d/vxscsiinq device_name
   ```

   where *device_name* is the device name of one of the disks in the array. Note the values of the vendor ID (VID) and product ID (PID) in the output from this command. For Fujitsu disks, also note the number of characters in the serial number that is displayed.

   The following example shows the output for the example disk with the device name `/dev/hdisk10`

   ```
   # /etc/vx/diag.d/vxscsiinq /dev/hdisk10
   ```

   ```
   Vendor id (VID)      : SEAGATE
   Product id (PID)     : ST318404LSUN18G
   Revision             : 8507
   Serial Number        : 0025T0LA3H
   ```

   In this example, the vendor ID is `SEAGATE` and the product ID is `ST318404LSUN18G`.

2. Stop all applications, such as databases, from accessing VxVM volumes that are configured on the array, and unmount all file systems and Storage Checkpoints that are configured on the array.

3. If the array is of type A/A-A, A/P or A/PF, configure it in autotrespass mode.

4 Enter the following command to add a new JBOD category:

```
# vxddladm addjbod vid=vendorid [pid=productid] \
[serialnum=opcode/pagecode/offset/length]
[cabinetnum=opcode/pagecode/offset/length] policy={aa|ap}]
```

where *vendorid* and *productid* are the VID and PID values that you found from the previous step. For example, *vendorid* might be `FUJITSU`, `IBM`, or `SEAGATE`. For Fujitsu devices, you must also specify the number of characters in the serial number as the argument to the `length` argument (for example, 10). If the array is of type A/A-A, A/P or A/PF, you must also specify the `policy=ap` attribute.

Continuing the previous example, the command to define an array of disks of this type as a JBOD would be:

```
# vxddladm addjbod vid=SEAGATE pid=ST318404LSUN18G
```

5 Use the `vxdctl enable` command to bring the array under VxVM control.

```
# vxdctl enable
```

See "Enabling discovery of new disk arrays" on page 137.

6 To verify that the array is now supported, enter the following command:

```
# vxddladm listjbod
```

The following is sample output from this command for the example array:

```
VID      PID          SerialNum             CabinetNum    Policy
                   (Cmd/PageCode/off/len)  (Cmd/PageCode/off/len)
==============================================================
SEAGATE ALL PIDs   18/-1/36/12           18/-1/10/11     Disk
SUN      SESS01     18/-1/36/12           18/-1/12/11     Disk
```

7 To verify that the array is recognized, use the `vxdmpadm listenclosure` command as shown in the following sample output for the example array:

```
# vxdmpadm listenclosure
ENCLR_NAME ENCLR_TYPE ENCLR_SNO STATUS    ARRAY_TYPE LUN_COUNT
===============================================================
Disk       Disk       DISKS     CONNECTED Disk       2
```

The enclosure name and type for the array are both shown as being set to `Disk`. You can use the `vxdisk list` command to display the disks in the array:

```
# vxdisk list
DEVICE   TYPE        DISK     GROUP      STATUS
Disk_0   auto:none   -        -          online invalid
Disk_1   auto:none   -        -          online invalid
...
```

8 To verify that the DMP paths are recognized, use the `vxdmpadm getdmpnode` command as shown in the following sample output for the example array:

```
# vxdmpadm getdmpnode enclosure=Disk
NAME   STATE   ENCLR-TYPE PATHS ENBL DSBL ENCLR-NAME
=====================================================
Disk_0 ENABLED Disk       2     2    0    Disk
Disk_1 ENABLED Disk       2     2    0    Disk
...
```

This shows that there are two paths to the disks in the array.

For more information, enter the command `vxddladm help addjbod`.

See the `vxddladm`(1M) manual page.

See the `vxdmpadm`(1M) manual page.

Removing disks from the DISKS category

To remove disks from the `DISKS` category

◆ Use the `vxddladm` command with the `rmjbod` keyword. The following example illustrates the command for removing disks which have the vendor id of SEAGATE:

```
# vxddladm rmjbod vid=SEAGATE
```

Foreign devices

DDL may not be able to discover some devices that are controlled by third-party drivers, such as those that provide multi-pathing or RAM disk capabilities. For these devices it may be preferable to use the multi-pathing capability that is provided by the third-party drivers for some arrays rather than using Dynamic Multi-Pathing (DMP). Such foreign devices can be made available as simple disks to VxVM by using the `vxddladm addforeign` command. This also has the effect of bypassing DMP for handling I/O. The following example shows how to add entries for block and character devices in the specified directories:

```
# vxddladm addforeign blockdir=/dev/foo/dsk \
  chardir=/dev/foo/rdsk
```

By default, this command suppresses any entries for matching devices in the OS-maintained device tree that are found by the autodiscovery mechanism. You can override this behavior by using the `-f` and `-n` options as described on the `vxddladm`(1M) manual page.

After adding entries for the foreign devices, use either the `vxdisk scandisks` or the `vxdctl enable` command to discover the devices as simple disks. These disks then behave in the same way as autoconfigured disks.

The foreign device feature was introduced in VxVM 4.0 to support non-standard devices such as RAM disks, some solid state disks, and pseudo-devices such as EMC PowerPath.

Foreign device support has the following limitations:

- A foreign device is always considered as a disk with a single path. Unlike an autodiscovered disk, it does not have a DMP node.
- It is not supported for shared disk groups in a clustered environment. Only standalone host systems are supported.
- It is not supported for Persistent Group Reservation (PGR) operations.
- It is not under the control of DMP, so enabling of a failed disk cannot be automatic, and DMP administrative commands are not applicable.
- Enclosure information is not available to VxVM. This can reduce the availability of any disk groups that are created using such devices.
- The I/O Fencing and Cluster File System features are not supported for foreign devices.

If a suitable ASL is available and installed for an array, these limitations are removed.

See “Third-party driver coexistence” on page 137.

Changing the disk-naming scheme

You can either use enclosure-based naming for disks or the operating system's naming scheme. DMP commands display device names according to the current naming scheme.

The default naming scheme is enclosure-based naming (EBN). When you use DMP with native volumes, the disk naming scheme must be EBN, the use_avid attribute must be on, and the persistence attribute must be set to yes.

To change the disk-naming scheme

- Select `Change the disk naming scheme` from the `vxdiskadm` main menu to change the disk-naming scheme that you want DMP to use. When prompted, enter `y` to change the naming scheme.

  Alternatively, you can change the naming scheme from the command line. Use the following command to select enclosure-based naming:

  ```
  # vxddladm set namingscheme=ebn [persistence={yes|no}] \
  [lowercase=yes|no] [use_avid=yes|no]
  ```

  Use the following command to select operating system-based naming:

  ```
  # vxddladm set namingscheme=osn [persistence={yes|no}] \
  [lowercase=yes|no]
  ```

  The optional `persistence` argument allows you to select whether the names of disk devices that are displayed by DMP remain unchanged after disk hardware has been reconfigured and the system rebooted. By default, enclosure-based naming is persistent. Operating system-based naming is not persistent by default.

  To change only the naming persistence without changing the naming scheme, run the `vxddladm set namingscheme` command for the current naming scheme, and specify the persistence attribute.

  By default, the names of the enclosure are converted to lowercase, regardless of the case of the name specified by the ASL. The enclosure-based device names are therefore in lower case. Set the `lowercase=no` option to suppress the conversion to lowercase.

  For enclosure-based naming, the `use_avid` option specifies whether the Array Volume ID is used for the index number in the device name. By default, `use_avid=yes`, indicating the devices are named as *enclosure_avid*. If `use_avid` is set to `no`, DMP devices are named as *enclosure_index*. The index number is assigned after the devices are sorted by LUN serial number.

  The change is immediate whichever method you use.

  See "Regenerating persistent device names" on page 152.

Displaying the disk-naming scheme

DMP disk naming can be operating-system based naming or enclosure-based naming. This command displays whether the DMP disk naming scheme is currently set. It also displays the attributes for the disk naming scheme, such as whether persistence is enabled.

To display the current disk-naming scheme and its mode of operations, use the following command:

```
# vxddladm get namingscheme
```

See "Disk device naming in DMP" on page 24.

Regenerating persistent device names

The persistent device naming feature makes the names of disk devices persistent across system reboots. DDL assigns device names according to the persistent device name database.

If operating system-based naming is selected, each disk name is usually set to the name of one of the paths to the disk. After hardware reconfiguration and a subsequent reboot, the operating system may generate different names for the paths to the disks. Therefore, the persistent device names may no longer correspond to the actual paths. This does not prevent the disks from being used, but the association between the disk name and one of its paths is lost.

Similarly, if enclosure-based naming is selected, the device name depends on the name of the enclosure and an index number. If a hardware configuration changes the order of the LUNs exposed by the array, the persistent device name may not reflect the current index.

To regenerate persistent device names

- To regenerate the persistent names repository, use the following command:

  ```
  # vxddladm [-c] assign names
  ```

  The `-c` option clears all user-specified names and replaces them with autogenerated names.

  If the `-c` option is not specified, existing user-specified names are maintained, but OS-based and enclosure-based names are regenerated.

  The disk names now correspond to the new path names.

Changing device naming for TPD-controlled enclosures

By default, TPD-controlled enclosures use pseudo device names based on the TPD-assigned node names. If you change the device naming to native, the devices are named in the same format as other DMP devices. The devices use either operating system names (OSN) or enclosure-based names (EBN), depending on which naming scheme is set.

See "Displaying the disk-naming scheme" on page 151.

To change device naming for TPD-controlled enclosures

- For disk enclosures that are controlled by third-party drivers (TPD) whose coexistence is supported by an appropriate ASL, the default behavior is to assign device names that are based on the TPD-assigned node names. You can use the `vxdmpadm` command to switch between these names and the device names that are known to the operating system:

  ```
  # vxdmpadm setattr enclosure enclosure_name tpdmode=native|pseudo
  ```

  The argument to the `tpdmode` attribute selects names that are based on those used by the operating system (`native`), or TPD-assigned node names (`pseudo`).

  The use of this command to change between TPD and operating system-based naming is illustrated in the following example for the enclosure named `EMC0`. In this example, the device-naming scheme is set to OSN.

  ```
  # vxdisk list

  DEVICE        TYPE           DISK      GROUP     STATUS
  emcpower10    auto:sliced    disk1     mydg      online
  emcpower11    auto:sliced    disk2     mydg      online
  emcpower12    auto:sliced    disk3     mydg      online
  emcpower13    auto:sliced    disk4     mydg      online
  emcpower14    auto:sliced    disk5     mydg      online
  emcpower15    auto:sliced    disk6     mydg      online
  emcpower16    auto:sliced    disk7     mydg      online
  emcpower17    auto:sliced    disk8     mydg      online
  emcpower18    auto:sliced    disk9     mydg      online
  emcpower19    auto:sliced    disk10    mydg      online

  # vxdmpadm setattr enclosure EMC0 tpdmode=native

  # vxdisk list

  DEVICE        TYPE           DISK      GROUP     STATUS
  hdisk1           auto:sliced    disk1     mydg      online
  hdisk2           auto:sliced    disk2     mydg      online
  hdisk3           auto:sliced    disk3     mydg      online
  hdisk4           auto:sliced    disk4     mydg      online
  hdisk5           auto:sliced    disk5     mydg      online
  hdisk6           auto:sliced    disk6     mydg      online
  hdisk7           auto:sliced    disk7     mydg      online
  hdisk8           auto:sliced    disk8     mydg      online
  hdisk9           auto:sliced    disk9     mydg      online
  hdisk10          auto:sliced    disk10    mydg      online
  ```

If `tpdmode` is set to `native`, the path with the smallest device number is displayed.

Discovering the association between enclosure-based disk names and OS-based disk names

If you enable enclosure-based naming, the `vxprint` command displays the structure of a volume using enclosure-based disk device names (disk access names) rather than OS-based names.

To discover the association between enclosure-based disk names and OS-based disk names

- ◆ To discover the operating system-based names that are associated with a given enclosure-based disk name, use either of the following commands:

  ```
  # vxdisk list enclosure-based_name
  # vxdmpadm getsubpaths dmpnodename=enclosure-based_name
  ```

  For example, to find the physical device that is associated with disk `ENC0_21`, the appropriate commands would be:

  ```
  # vxdisk list ENC0_21
  # vxdmpadm getsubpaths dmpnodename=ENC0_21
  ```

  To obtain the full pathname for the block disk device and the character disk device from these commands, append the displayed device name to `/dev/vx/dmp` or `/dev/vx/rdmp`.

Chapter 6

Online dynamic reconfiguration

This chapter includes the following topics:

About online dynamic reconfiguration

You can perform the following kinds of online dynamic reconfigurations:

- Reconfiguring a LUN online that is under DMP control
- Updating the array controller firmware, also known as a nondisruptive upgrade
- Replacing a host bus adapter (HBA) online

Reconfiguring a LUN online that is under DMP control

System administrators and storage administrators may need to modify the set of LUNs provisioned to a server. You can change the LUN configuration dynamically, without performing a reconfiguration reboot on the host.

Dynamic LUN reconfigurations require array configuration commands, operating system commands, and Veritas Volume manager commands. To complete the operations correctly, you must issue the commands in the proper sequence on the host.

The operations are as follows:

- Dynamic LUN removal from an existing target ID
  See "Removing LUNs dynamically from an existing target ID" on page 156.
- Dynamic new LUN addition to a new target ID
  See "Adding new LUNs dynamically to a new target ID" on page 158.
- Changing the LUN characteristics
  See "Changing the characteristics of a LUN from the array side" on page 160.

Removing LUNs dynamically from an existing target ID

In this case, a group of LUNs is unmapped from the host HBA ports and an operating system device scan is issued. To add subsequent LUNs seamlessly, perform additional steps to cleanup the operating system device tree.

The high-level procedure and the DMP commands are generic.

To remove LUNs dynamically from an existing target ID

1. Identify which LUNs to remove from the host. Do one of the following:
   - Use Storage Array Management to identify the Array Volume ID (AVID) for the LUNs.
   - If the array does not report the AVID, use the LUN index.
2. For LUNs under VxVM, perform the following steps:
   - Evacuate the data from the LUNs using the `vxevac` command.
     See the `vxevac`(1M) online manual page.
     After the data has been evacuated, enter the following command to remove the LUNs from the disk group:

     ```
     # vxdg -g diskgroup rmdisk da-name
     ```

   - If the data has not been evacuated and the LUN is part of a subdisk or disk group, enter the following command to remove the LUNs from the disk group. If the disk is part of a shared disk group, you must use the `-k` option to force the removal.

     ```
     # vxdg -g diskgroup -k rmdisk da-name
     ```

3. For LUNs using AIX LVM over DMP devices, remove the device from the LVM volume group.

   ```
   # reducevg vgname pvname
   ```

4 Using the AVID or LUN index, use Storage Array Management to unmap or unmask the LUNs you identified in step 1.

5 Remove the LUNs from the `vxdisk` list. Enter the following command on all nodes in a cluster:

```
# vxdisk rm da-name
```

This is a required step. If you do not perform this step, the DMP device tree shows ghost paths.

6 Clean up the AIX SCSI device tree for the devices that you removed in step 5.

See "Cleaning up the operating system device tree after removing LUNs" on page 159.

This step is required. You must clean up the operating system SCSI device tree to release the SCSI target ID for reuse if a new LUN is added to the host later.

7 Scan the operating system device tree.

See "Scanning an operating system device tree after adding or removing LUNs" on page 159.

8 Use DMP to perform a device scan. You must perform this operation on all nodes in a cluster. Enter one of the following commands:

- `# vxdctl enable`
- `# vxdisk scandisks`

9 Refresh the DMP device name database using the following command:

```
# vxddladm assign names
```

10 Verify that the LUNs were removed cleanly by answering the following questions:

- Is the device tree clean?
  After the LUN is removed cleanly, there should be no `hdisk` entries in the "Defined" state.
- Were all the appropriate LUNs removed?
  Use the DMP disk reporting tools such as the `vxdisk list` command output to determine if the LUNs have been cleaned up successfully.
- Is the `vxdisk list` output correct?

Verify that the `vxdisk list` output shows the correct number of paths and does not include any ghost disks.

If the answer to any of these questions is "No," return to step 4 and perform the required steps.

If the answer to all of the questions is "Yes," the LUN remove operation is successful.

Adding new LUNs dynamically to a new target ID

In this case, a new group of LUNs is mapped to the host via multiple HBA ports. An operating system device scan is issued for the LUNs to be recognized and added to DMP control.

The high-level procedure and the DMP commands are generic.

To add new LUNs dynamically to a new target ID

1. Identify which LUNs to add to the host. Do one of the following:
   - Use Storage Array Management to identify the Array Volume ID (AVID) for the LUNs.
   - If the array does not report the AVID, use the LUN index.
2. Map/mask the LUNs to the new target IDs on multiple hosts.
3. Scan the operating system device.

   See "Scanning an operating system device tree after adding or removing LUNs" on page 159.

   Repeat step 1 and step 2 until you see that all the LUNs have been added.
4. Use DMP to perform a device scan. You must perform this operation on all nodes in a cluster. Enter one of the following commands:
   - `# vxdctl enable`
   - `# vxdisk scandisks`
5. Refresh the DMP device name database using the following command:

   ```
   # vxddladm assign names
   ```
6. Verify that the LUNs were added correctly by answering the following questions:
   - Do the newly provisioned LUNs appear in the `vxdisk list` output?

- Are the configured paths present for each LUN?

If the answer to any of these questions is "No," return to step 1 and begin the procedure again.

If the answer to all of the questions is "Yes," the LUNs have been successfully added. You can now add the LUNs to a disk group, create new volumes, or grow existing volumes.

If the dmp_native_support tunable is set to ON and the new LUN does not have a VxVM label or is not claimed by a TPD driver then the LUN is available for use by LVM.

About detecting target ID reuse if the operating system device tree is not cleaned up

When the target ID is reused and the operating system device tree is not cleaned up, the `vxdisk scandisks` and `vxdctl enable` commands hang. To correct this situation, you must clean up the operating system device tree.

See "Cleaning up the operating system device tree after removing LUNs" on page 159.

Scanning an operating system device tree after adding or removing LUNs

After you add or remove LUNs, scan the operating system device tree to verify that the operation completed successfully.

To scan an operating system device tree after adding or removing LUNs

- Enter the following command:

```
# cfgmgr -v
```

Cleaning up the operating system device tree after removing LUNs

After you remove LUNs, you must clean up the operating system device tree.

To clean up the operating system device tree after removing LUNs

1. Enter the following command. Devices that have been removed will have `Defined` after the disk name.

```
# lsdev -Cc disk
hdisk431 Defined  09-08-02 IBM 2810XIV Non-MPIO Fibre Channel Disk
hdisk432 Defined  0A-08-02 IBM 2810XIV Non-MPIO Fibre Channel Disk
```

2. For each disk name, run the following command to remove it from the operating system database:

```
# rmdev -dl hdisk-name
```

In this example, the commands would be the following:

```
# rmdev -dl hdisk431
```

```
# rmdev -dl hdisk432
```

3. Repeat step 1 and verify that no devices are shown as `Defined`.

Changing the characteristics of a LUN from the array side

Some arrays provide a way to change the properties of LUNs. For example, the EMC symmetrix array allows write-protected (Read-only), and read-write enabled LUNs. Before changing the properties of a LUN, you must remove the device from Veritas Volume Manager (VxVM) control.

To change the properties of a LUN

1. If the device is part of a disk group, move the disk out of the disk group.

```
# vxdg -g dgname rmdisk daname
```

2. Remove the disk from the `vxdisk` list.

In a cluster, perform this step from all of the nodes.

```
# vxdisk rm da-name
```

For example:

```
# vxdisk rm eva4k6k0_0
```

3. Change the device characteristics.

4 Use DMP to perform a device scan. In a cluster, perform this command on all the nodes.

```
# vxdisk scandisks
```

5 Add the device back to the disk group.

```
# vxdg -g dgname adddisk daname
```

Upgrading the array controller firmware online

Storage array subsystems need code upgrades as fixes, patches, or feature upgrades. You can perform these upgrades online when the file system is mounted and I/Os are being served to the storage.

Legacy storage subsystems contain two controllers for redundancy. An online upgrade is done one controller at a time. DMP fails over all I/O to the second controller while the first controller is undergoing an Online Controller Upgrade. After the first controller has completely staged the code, it reboots, resets, and comes online with the new version of the code. The second controller goes through the same process, and I/O fails over to the first controller.

Note: Throughout this process, application I/O is not affected.

Array vendors have different names for this process. For example, EMC calls it a nondisruptive upgrade (NDU) for CLARiiON arrays.

A/A type arrays require no special handling during this online upgrade process. For A/P, A/PF, and ALUA type arrays, DMP performs array-specific handling through vendor-specific array policy modules (APMs) during an online controller code upgrade.

When a controller resets and reboots during a code upgrade, DMP detects this state through the SCSI Status. DMP immediately fails over all I/O to the next controller.

If the array does not fully support NDU, all paths to the controllers may be unavailable for I/O for a short period of time. Before beginning the upgrade, set the `dmp_lun_retry_timeout` tunable to a period greater than the time that you expect the controllers to be unavailable for I/O. DMP retries the I/Os until the end of the `dmp_lun_retry_timeout` period, or until the I/O succeeds, whichever happens first. Therefore, you can perform the firmware upgrade without interrupting the application I/Os.

For example, if you expect the paths to be unavailable for I/O for 300 seconds, use the following command:

```
# vxdmpadm settune dmp_lun_retry_timeout=300
```

DMP retries the I/Os for 300 seconds, or until the I/O succeeds.

To verify which arrays support Online Controller Upgrade or NDU, see the hardware compatibility list (HCL) at the following URL:

http://entsupport.symantec.com/docs/330441

Replacing a host bus adapter online

Before you replace a host bus adapter (HBA) online, you must disable the I/O paths to the controller. After you replace the HBA, you enable the I/O paths.

To replace a host bus adapter online

1. Disable the paths to the controller by removing the reference from DMP. Enter the following. In this example, the controller name is `fscsi`.

   ```
   # vxdmpadm -f disable ctlr=fscsi
   ```

2. Remove the device references from the operating system. Enter the following:

   ```
   # rmdev -Rdl fscsi
   ```

3. Rescan the device tree and rebuild the DMP database. Enter the following:

   ```
   # vxdctl enable
   ```

4. Replace the host bus adapter.
5. Reconfigure the devices in the operating system. Enter the following:

   ```
   # cfgmgr
   ```

6. Verify that new devices appear at the operating system level.

   ```
   # lsdev -Cc disk
   ```

7 Enable the controller. In this example, the controller name is `fscsi`.

```
# vxdmpadm enable ctlr=fscsi
```

8 Rescan the device tree and rebuild the DMP database.

```
# vxdctl enable
```

Chapter 7

Event monitoring

This chapter includes the following topics:

About the event source daemon (vxesd)

The event source daemon (`vxesd`) is a Veritas Dynamic Multi-Pathing (DMP) component process that receives notifications of any device-related events that are used to take appropriate actions. The benefits of `vxesd` include:

- Monitoring of SAN fabric events and proactive error detection (SAN event)
- Logging of DMP events for troubleshooting (DMP event)
- Discovery of SAN components and HBA-array port connectivity (Fibre Channel and iSCSI)

Fabric Monitoring and proactive error detection

In previous releases, DMP handled failed paths reactively, by only disabling paths when active I/O failed on the storage. Using the Storage Networking Industry Association (SNIA) HBA API library, `vxesd` now is able to receive SAN fabric events from the HBA. This information allows DMP to take a proactive role by checking suspect devices from the SAN events, even if there is no active I/O. New I/O is directed to healthy paths while the suspect devices are verified.

During startup, `vxesd` queries the HBA (by way of the SNIA library) to obtain the SAN topology. The `vxesd` daemon determines the Port World Wide Names (PWWN) that correspond to each of the device paths that are visible to the operating system. After the `vxesd` daemon obtains the topology, `vxesd` registers with the HBA for SAN event notification. If LUNs are disconnected from a SAN, the HBA notifies `vxesd` of the SAN event, specifying the PWWNs that are affected. The `vxesd` daemon uses this event information and correlates it with the previous topology information to determine which set of device paths have been affected.

The `vxesd` daemon sends the affected set to the `vxconfigd` daemon (DDL) so that the device paths can be marked as suspect. When the path is marked as suspect, DMP does not send new I/O to the path unless it is the last path to the device. In the background, the DMP restore daemon checks the accessibility of the paths on its next periodic cycle using a SCSI inquiry probe. If the SCSI inquiry fails, DMP disables the path to the affected LUNs, which is also logged in the event log.

If the LUNs are reconnected at a later time, the HBA informs `vxesd` of the SAN event. When the DMP restore daemon runs its next test cycle, the disabled paths are checked with the SCSI probe and re-enabled if successful.

Note: If `vxesd` receives an HBA LINK UP event, the DMP restore daemon is restarted and the SCSI probes run immediately, without waiting for the next periodic cycle. When the DMP restore daemon is restarted, it starts a new periodic cycle. If the disabled paths are not accessible by the time of the first SCSI probe, they are re-tested on the next cycle (300s by default).

The fabric monitor functionality is enabled by default. The value of the `dmp_monitor_fabric` tunable is persistent across reboots.

To disable the Fabric Monitoring functionality, use the following command:

```
# vxdmpadm settune dmp_monitor_fabric=off
```

To enable the Fabric Monitoring functionality, use the following command:

```
# vxdmpadm settune dmp_monitor_fabric=on
```

To display the current value of the `dmp_monitor_fabric` tunable, use the following command:

```
# vxdmpadm gettune dmp_monitor_fabric
```

Discovery of iSCSI and SAN Fibre Channel topology

The vxesd builds a topology of iSCSI and Fibre Channel devices that are visible to the host. The vxesd daemon uses the SNIA Fibre Channel HBA API to obtain the SAN topology. If IMA is not available, then iSCSI management CLI is used to obtain the iSCSI SAN topology.

To display the hierarchical listing of Fibre Channel and iSCSI devices, use the following command:

```
# vxddladm list
```

See the vxddladm(1M) manual page.

DMP event logging

DMP notifies vxesd of major events, and vxesd logs the event in a log file (/etc/vx/dmpevents.log). These events include:

- Marking paths or dmpnodes enabled
- Marking paths or dmpnodes disabled
- Throttling of paths i/o error analysis HBA/SAN events

The log file is located in /var/adm/vx/dmpevents.log but is symbolically linked to /etc/vx/dmpevents.log. When the file reaches 10,000 lines, the log is rotated. That is, dmpevents.log is renamed dmpevents.log.X and a new dmpevents.log file is created.

You can change the level of detail in the event log file using the tunable dmp_log_level. Valid values are 1 through 4.

```
# vxdmpadm settune dmp_log_level=X
```

The current value of dmp-log_level can be displayed with:

```
# vxdmpadm gettune dmp_log_level
```

For details on the various log levels, see the vxdmpadm(1M) manual page.

Starting and stopping the event source daemon

By default, DMP starts vxesd at boot time.

To stop the vxesd daemon, use the vxddladm utility:

```
# vxddladm stop eventsource
```

To start the `vxesd` daemon, use the `vxddladm` utility:

```
# vxddladm start eventsource [logfile=logfilename]
```

Chapter 8

Performance monitoring and tuning

This chapter includes the following topics:

Configuring the AIX fast fail feature for use with Veritas Volume Manager (VxVM) and Dynamic Multi-Pathing (DMP)

DMP failover takes significant time when the path is disabled from the switch or array side in a SAN environment. This issue is not seen if the path is disabled from the host side. The dynamic tracking and fast fail features of AIX prevent the long failover time.

To configure the AIX fast fail feature for use with VxVM and DMP

1 Enter the following commands for each Fibre Channel adapter or controller:

```
# chdev -l fscsiN -a fc_err_recov=fast_fail -P
# chdev -l fscsiN -a dyntrk=yes -P
```

where *N* is the number of the controller (0, 1, 2 and so on).

2 Reboot the system.

3 Use the `lsattr` command to verify that the dyntrk and fast_fail attributes are set to True on each adapter, as shown in this example:

```
# lsattr -El fscsi0
attach       switch    How this adapter is CONNECTED       False
dyntrk       yes       Dynamic Tracking of FC Devices      True
fc_err_recov fast_fail FC Fabric Event Error Recovery Policy True
scsi_id      0x10d00   Adapter SCSI ID                     False
sw_fc_class  3         FC Class for Fabric controllers.    True
```

DMP tunable parameters

DMP tunables are set online (without requiring a reboot) by using the `vxdmpadm` command as shown here:

```
# vxdmpadm settune dmp_tunable=value
```

The values of these tunables can be displayed by using this command:

```
# vxdmpadm gettune [dmp_tunable]
```

Table 8-1 shows the DMP parameters that can be tuned by using the `vxdmpadm settune` command.

Table 8-1 DMP parameters that are tunable

Parameter	Description
`dmp_cache_open`	If this parameter is set to `on`, the first open of a device that is performed by an array support library (ASL) is cached. This caching enhances the performance of device discovery by minimizing the overhead that is caused by subsequent opens by ASLs. If this parameter is set to `off`, caching is not performed. The default value is `on`.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
dmp_daemon_count	The number of kernel threads that are available for servicing path error handling, path restoration, and other DMP administrative tasks. The default number of threads is 10.
dmp_delayq_interval	How long DMP should wait before retrying I/O after an array fails over to a standby path. Some disk arrays are not capable of accepting I/O requests immediately after failover. The default value is 15 seconds.
dmp_enable_restore	If this parameter is set to `on`, it enables the path restoration thread to be started. See "Configuring DMP path restoration policies" on page 129. If this parameter is set to `off`, it disables the path restoration thread. If the path restoration thread is currently running, use the `vxdmpadm stop restore` command to stop the thread. The default is `on`. See "Stopping the DMP path restoration thread" on page 130.
dmp_fast_recovery	Whether DMP should try to obtain SCSI error information directly from the HBA interface. Setting the value to `on` can potentially provide faster error recovery, provided that the HBA interface supports the error enquiry feature. If this parameter is set to `off`, the HBA interface is not used. The default setting is `on`.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_health_time`	DMP detects intermittently failing paths, and prevents I/O requests from being sent on them. The value of `dmp_health_time` represents the time in seconds for which a path must stay healthy. If a path's state changes back from enabled to disabled within this time period, DMP marks the path as intermittently failing, and does not re-enable the path for I/O until `dmp_path_age` seconds elapse. The default value is 60 seconds. A value of 0 prevents DMP from detecting intermittently failing paths.
`dmp_log_level`	The level of detail that is displayed for DMP console messages. The following level values are defined: 1 — Displays all DMP log messages that existed in releases before 5.0. 2 — Displays level 1 messages plus messages that relate to path or disk addition or removal, SCSI errors, IO errors and DMP node migration. 3 — Displays level 1 and 2 messages plus messages that relate to path throttling, suspect path, idle path and insane path logic. 4 — Displays level 1, 2 and 3 messages plus messages that relate to setting or changing attributes on a path and tunable related changes. The default value is 1.
`dmp_low_impact_probe`	Determines if the path probing by restore daemon is optimized or not. Set it to `on` to enable optimization and `off` to disable. Path probing is optimized only when restore policy is check_disabled or during check_disabled phase of check_periodic policy. The default value is `on`.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_lun_retry_timeout`	Retry period for handling transient errors. The value is specified in seconds. When all paths to a disk fail, there may be certain paths that have a temporary failure and are likely to be restored soon. The I/Os may be failed to the application layer even though the failures are transient, unless the I/Os are retried. The `dmp_lun_retry_timeout` tunable provides a mechanism to retry such transient errors. If the tunable is set to a non-zero value, I/Os to a disk with all failed paths are retried until `dmp_lun_retry_timeout` interval or until the I/O succeeds on one of the path, whichever happens first. The default value of tunable is 0, which means that the paths are probed only once.
`dmp_monitor_fabric`	Determines whether the Event Source daemon (`vxesd`) uses the Storage Networking Industry Association (SNIA) HBA API. This API allows DDL to improve the performance of failover by collecting information about the SAN topology and by monitoring fabric events. If this parameter is set to `on`, DDL uses the SNIA HBA API. (Note that the HBA vendor specific HBA-API library should be available to use this feature.) If this parameter is set to `off`, the SNIA HBA API is not used. The default setting is `off`. Symantec recommends that this setting remain off to avoid performance issues on the AIX platform.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_monitor_osevent`	Determines whether the Event Source daemon (`vxesd`) monitors operating system events such as reconfiguration operations. If this parameter is set to `on`, `vxesd` monitors operations such as attaching operating system devices. If this parameter is set to `off`, `vxesd` does not monitor operating system operations. When DMP co-exists with EMC PowerPath, Symantec recommends setting this parameter to `off` to avoid any issues. The default setting is `on`, unless EMC PowerPath is installed. If you install DMP on a system that already has PowerPath installed, DMP sets the `dmp_monitor_osevent` to `off`.
`dmp_native_support`	Determines whether DMP will do multi-pathing for native devices. Set the tunable to `on` to have DMP do multi-pathing for native devices. When a Storage Foundation product is installed, the default value is `off`. When Veritas Dynamic Multi-Pathing is installed, the default value is `on`.
`dmp_path_age`	The time for which an intermittently failing path needs to be monitored as healthy before DMP again tries to schedule I/O requests on it. The default value is 300 seconds. A value of 0 prevents DMP from detecting intermittently failing paths.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_pathswitch_blks_shift`	The default number of contiguous I/O blocks that are sent along a DMP path to an array before switching to the next available path. The value is expressed as the integer exponent of a power of 2; for example 9 represents 512 blocks. The default value of this parameter is set to 9. In this case, 512 blocks (256k) of contiguous I/O are sent over a DMP path before switching. For intelligent disk arrays with internal data caches, better throughput may be obtained by increasing the value of this tunable. For example, for the HDS 9960 A/A array, the optimal value is between 15 and 17 for an I/O activity pattern that consists mostly of sequential reads or writes. This parameter only affects the behavior of the `balanced` I/O policy. A value of 0 disables multi-pathing for the policy unless the `vxdmpadm` command is used to specify a different partition size for an array. See "Specifying the I/O policy" on page 115.
`dmp_probe_idle_lun`	If DMP statistics gathering is enabled, set this tunable to `on` (default) to have the DMP path restoration thread probe idle LUNs. Set this tunable to `off` to turn off this feature. (Idle LUNs are VM disks on which no I/O requests are scheduled.) The value of this tunable is only interpreted when DMP statistics gathering is enabled. Turning off statistics gathering also disables idle LUN probing. The default value is `on`.
`dmp_probe_threshold`	If the dmp_low_impact_probe is turned `on`, dmp_probe_threshold determines the number of paths to probe before deciding on changing the state of other paths in the same subpath failover group. The default value is 5.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_queue_depth`	The maximum number of queued I/O requests on a path during I/O throttling. The default value is 32. A value can also be set for paths to individual arrays by using the `vxdmpadm` command. See "Configuring the I/O throttling mechanism" on page 126.
`dmp_restore_cycles`	If the DMP restore policy is `check_periodic`, the number of cycles after which the `check_all` policy is called. The default value is 10. The value of this tunable can also be set using the `vxdmpadm start restore` command. See "Configuring DMP path restoration policies" on page 129.
`dmp_restore_interval`	The interval attribute specifies how often the path restoration thread examines the paths. Specify the time in seconds. The default value is 300. The value of this tunable can also be set using the `vxdmpadm start restore` command. See "Configuring DMP path restoration policies" on page 129.
`dmp_restore_policy`	The DMP restore policy, which can be set to one of the following values: ■ `check_all` ■ `check_alternate` ■ `check_disabled` ■ `check_periodic` The default value is `check_disabled`. The value of this tunable can also be set using the `vxdmpadm start restore` command. See "Configuring DMP path restoration policies" on page 129.

Table 8-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_retry_count`	If an inquiry succeeds on a path, but there is an I/O error, the number of retries to attempt on the path. The default value is 5. A value can also be set for paths to individual arrays by using the `vxdmpadm` command. See "Configuring the response to I/O failures" on page 124.
`dmp_scsi_timeout`	Determines the timeout value to be set for any SCSI command that is sent via DMP. If the HBA does not receive a response for a SCSI command that it has sent to the device within the timeout period, the SCSI command is returned with a failure error code. The default value is 30 seconds.
`dmp_sfg_threshold`	Determines the minimum number of paths that should be failed in a failover group before DMP starts suspecting other paths in the same failover group. The value of 0 disables the failover logic based on subpath failover groups. The default value is 1.
`dmp_stat_interval`	The time interval between gathering DMP statistics. The default and minimum value are 1 second.

DMP driver tunables

DMP uses a slab allocator to service I/Os. DMP uses the DMP driver tunables `dmpslab_minsz` and `dmpslab_maxsz` to control the memory allocated for this slab allocator. These tunables are defined as follows:

dmpslab_maxsz	Maximum size of the slab.The size is specified in pages, where 1 page equals 4096 bytes. The default value for dmpslab_maxsz is 5% of the physical memory.

dmpslab_minsz	The minimum memory size that should be allocated to the slab during the driver load time. The size is specified in pages, where 1 page equals 4096 bytes. The default value for dmpslab_mazsz is 24 pages.

To display the tunables, use the following command:

```
# lsattr -El vxdmp
dmpslab_maxsz   101580 N/A True
dmpslab_minsz   32     N/A True
```

Note: If the errpt displays ENOMEM error code, you might need to change the dmpslab_minsz and dmpslab_maxsz to suit the load on the system.

Changing the value of the DMP driver tunables

1. Specify a new size in pages. You must increase the size in multiples of 8.

   To change the dmpslab_minsz tunable:

   ```
   # chdev -P -l vxdmp -a dmpslab_minsz=newsize
   ```

   To change the dmpslab_maxsz tunable:

   ```
   # chdev -P -l vxdmp -a dmpslab_maxsz=newsize
   ```

2. Reboot the system for the new values to take effect.

DMP troubleshooting

This appendix includes the following topics:

Displaying extended attributes after upgrading to DMP 5.1SP1

You may see the following changes in functionality from the Storage Foundation 5.1 release:

- The device names that are listed in the `vxdisk list` output do not display the Array Volume IDs (AVIDs).
- The `vxdisk -e list` output does not display extended attributes.
- An Active/Passive (A/P) or ALUA array is claimed as Active/Active (A/A).

This behavior may be because the LUNs are controlled by the native multi-pathing driver, MPIO.

To check whether LUNs are controlled by native multi-pathing driver

◆ Check the output of the following command to see if the LUN is an MPIO device:

```
# lsdev -Cc disk
```

You can migrate the LUNs from the control of the native multi-pathing driver to DMP control.

- To migrate to DMP with Veritas Volume Manager, refer to the section on disabling MPIO in the *Veritas Volume Manager Administrator's Guide*.

- To migrate to DMP with OS native volume support, refer to the section on migrating to DMP from MPIO in the *Veritas Dynamic Multi-Pathing Adminstrator's Guide*.

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.

associate — The process of establishing a relationship between VxVM 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.

In a cluster, an atomic operation takes place either on all nodes or not at all.

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 or from a disk or array.

boot disk — A disk that is used for the purpose of booting a system.

boot disk group — A private disk group that contains the disks from which the system may be booted.

bootdg — A reserved disk group name that is an alias for the name of the boot disk group.

clean node shutdown — The ability of a node to leave a cluster gracefully when all access to shared volumes has ceased.

cluster — A set of hosts (each termed a node) 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 access to the disks is 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 copy A single copy of a configuration database.

configuration database A set of records containing detailed information on existing VxVM objects (such as disk and volume attributes).

DCO (data change object) A VxVM object that is used to manage information about the FastResync maps in the DCO volume. Both a DCO object and a DCO volume must be associated with a volume to implement Persistent FastResync on that volume.

data stripe This represents the usable data portion of a stripe and is equal to the stripe minus the parity region.

DCO volume A special volume that is used to hold Persistent FastResync change maps and dirty region logs. See also see dirty region logging.

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 `hdisk3`, which indicates the whole of disk 3.

In a SAN environment, it is more convenient to use enclosure-based naming, which forms the device name by concatenating the name of the enclosure (such as `enc0`) with the disk's number within the enclosure, separated by an underscore (for example, `enc0_2`). The term disk access name can also be used to refer to a device name.

dirty region logging The method by which the VxVM monitors and logs modifications to a plex as a bitmap of changed regions. For a volumes with a new-style DCO volume, the dirty region log (DRL) is maintained in the DCO volume. Otherwise, the DRL is allocated to 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 An alternative term for a device 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 VxVM 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 In the multipathing subsystem of VxVM, the controller (host bus adapter or HBA) or disk array connected to the host, which the operating system represents as the parent node of a disk.

disk enclosure An intelligent disk array that usually has a backplane with a built-in Fibre Channel loop, and which permits hot-swapping of disks.

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 VxVM 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 disk group names `bootdg` (an alias for the boot disk group), `defaultdg` (an alias for the default disk group) and `nodg` (represents no disk group) are reserved.

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 An alternative term for a disk name.

disk media record A configuration record that identifies a particular disk, by disk ID, and gives that disk a logical (or administrative) name.

disk name A logical or administrative name chosen for a disk that is under the control of VxVM, such as `disk03`. The term disk media name is also used to refer to a disk name.

dissociate The process by which any link that exists between two VxVM 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 in a cluster, 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.

Encapsulation is not supported on the AIX platform.

enclosure See disk enclosure.

enclosure-based naming See device name.

fabric mode disk A disk device that is accessible on a Storage Area Network (SAN) via a Fibre Channel switch.

FastResync A fast resynchronization feature that is used to perform quick and efficient resynchronization of stale mirrors, and to increase the efficiency of the snapshot mechanism.

Fibre Channel A collective name for the fiber optic technology that is commonly used to set up a Storage Area Network (SAN).

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 VxVM 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 VxVM. The host ID 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.

hot-swap Refers to devices that can be removed from, or inserted into, a system without first turning off the power supply to the system.

initiating node The node on which the system administrator is running a utility that requests a change to VxVM objects. This node initiates a volume reconfiguration.

JBOD (just a bunch of disks) The common name for an unintelligent disk array which may, or may not, support the hot-swapping of disks.

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.

master node A node that is designated by the software to coordinate certain VxVM operations in a cluster. Any node is capable of being the master node.

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 consists of one plex of the volume with which the mirror is associated.

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, (for example, 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.

Non-Persistent FastResync A form of FastResync that cannot preserve its maps across reboots of the system because it stores its change map in memory.

object An entity that is defined to and recognized internally by VxVM. 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/O for that disk onto the remaining (alternate) paths.

pathgroup In the case of disks which are not multipathed by `vxdmp`, VxVM will see each path as a disk. In such cases, all paths to the disk can be grouped. This way only one of the paths from the group is made visible to VxVM.

Persistent FastResync A form of FastResync that can preserve its maps across reboots of the system by storing its change map in a DCO volume on disk).

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 VxVM control.

plex A plex is a logical grouping of subdisks that creates an area of disk space independent of physical disk size or other restrictions. Mirroring is set up by creating multiple data plexes for a single volume. Each data plex in a mirrored volume contains an identical copy of the volume data. Plexes may also be created to represent concatenated, striped and RAID-5 volume layouts, and to store volume logs.

primary path In Active/Passive 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.

private disk group A disk group in which the disks are accessed by only one specific host in a cluster.

private region A region of a physical disk used to store private, structured VxVM 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 VxVM that contains available space and is used for allocating subdisks.

RAID (redundant array of independent disks) 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 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 VxVM control. The resulting volumes can then be mirrored to provide redundancy and allow recovery in the event of disk failure.

Rootability is not supported on the AIX platform.

secondary path In Active/Passive 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.

sector A unit of size, which can vary between systems. Sector size is set per device (hard drive, CD-ROM, and so on). Although all devices within a system are usually configured to the same sector size for interoperability, this is not always the case.

A sector is commonly 512 bytes.

shared disk group A disk group in which access to the disks is 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 of a cluster at the same time.

shared VM disk A VM disk that belongs to a shared disk group in a cluster.

slave node A node that is not designated as the master node of a cluster.

slice The standard division of a logical disk device. The terms partition and slice are sometimes used synonymously.

snapshot A point-in-time copy of a volume (volume snapshot) or a file system (file system snapshot).

spanning A layout technique that permits a volume (and its file system or database) that is too large to fit on a single disk to be configured 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 do not have a backing subdisk).

SAN (storage area network) A networking paradigm that provides easily reconfigurable connectivity between any subset of computers, disk storage and interconnecting hardware such as switches, hubs and bridges.

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 64KB. The stripe unit size is sometimes also referred to as the 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.

VM disk A disk that is both under VxVM control and assigned to a disk group. VM disks are sometimes referred to as VxVM disks.

volboot file A small file that is used to locate copies of the boot disk group configuration. The file may list disks that contain configuration copies in standard locations, and can also contain direct pointers to configuration copy locations. The `volboot` file is stored in a system-dependent location.

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`.

vxconfigd The VxVM configuration daemon, which is responsible for making changes to the VxVM configuration. This daemon must be running before VxVM operations can be performed.

Index