Dynamic Multi-Pathing 8.0.2 Administrator's Guide - Linux

VERITAS™

Last updated: 2023-06-05

Legal Notice

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054
http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:
https://www.veritas.com/support

You can manage your Veritas account information at the following URL:
https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:
https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:
infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:
http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:
https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Chapter 1

Understanding DMP

This chapter includes the following topics:

About Dynamic Multi-Pathing (DMP)

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

DMP metadevices support the OS native logical volume manager (LVM). You can create LVM volumes and volume groups on DMP metadevices.

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

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

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 87.

The multi-pathing policy that DMP uses depends on the characteristics of the disk array.

DMP supports the following standard array types:

Table 1-1

Array type	Description
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. The behavior is similar to ALUA, except that it does not support the SCSI commands that an ALUA array supports.
Asymmetric Logical Unit Access (ALUA)	DMP supports all variants of ALUA.

Table 1-1 *(continued)*

Array type	Description
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 array mode 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/PF)	The appropriate command must be issued to the array to make the LUNs fail over to the secondary path. This array mode 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.

Table 1-1 *(continued)*

Array type	Description
Active/Passive with LUN group failover (A/PG)	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 array mode 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.

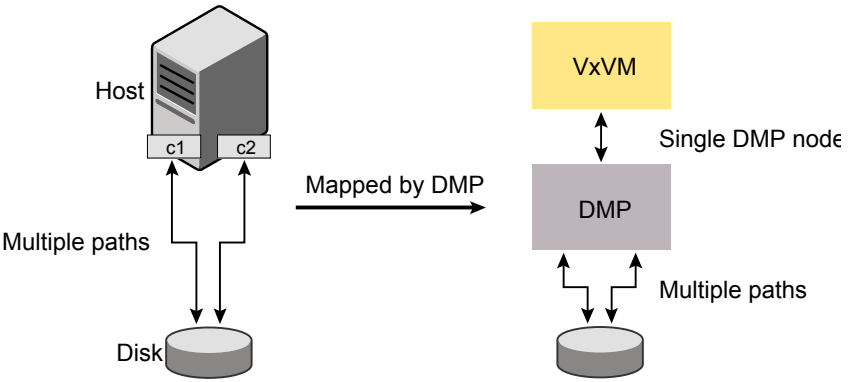
An array policy module (APM) may define array types to DMP in addition to the standard types for the arrays that it supports.

Dynamic Multi-Pathing 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.

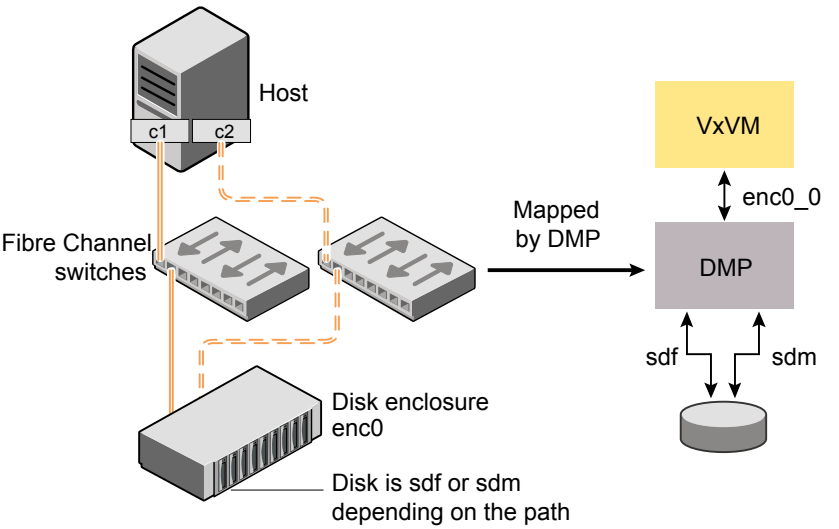
Figure 1-1 How DMP represents multiple physical paths to a disk as one node



DMP 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, `sdf` and `sdm`, 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 19.

See “Changing the disk device naming scheme” on page 104.

See "Discovering and configuring newly added disk devices" on page 87.

How DMP monitors I/O on paths

In VxVM prior to release 5.0, 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 name `restored` has 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 (by default, 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 82.

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 13.

See "I/O throttling" on page 14.

Path failover mechanism

DMP enhances system availability when used with disk arrays having multiple paths. 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 77.

Subpaths Failover Group (SFG)

A subpaths failover group (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, 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 I/O performance. Currently the criteria followed by DMP to form the subpaths 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 80.

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 when no I/O occurs on a path. 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 a limited number of paths within a subpaths failover group (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 (LIPP)" on page 80.

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 79.

Load balancing

By default, DMP uses the Minimum Queue I/O policy for load balancing across paths for all array types. Load balancing maximizes I/O throughput by using the total bandwidth of all available paths. I/O is sent down the path that has the minimum outstanding I/Os.

For Active/Passive (A/P) disk arrays, I/O is sent down the primary paths. If all of 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 other arrays, load balancing is performed across all the currently active paths.

You can change the I/O policy for the paths to an enclosure or disk array. This operation is an online operation that does not impact the server or require any downtime.

DMP in a clustered environment

In a clustered environment where Active/Passive (A/P) type disk arrays are shared by multiple hosts, all nodes in the cluster must access the disk through the same physical storage controller port. Accessing a disk through 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 Array Support Library (ASL) is installed on the system. An Array Policy Module (APM) may also be required.

See "About discovering disks and dynamically adding disk arrays" on page 89.

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 need to access a disk through the same physical path.

See "How to administer the Device Discovery Layer" on page 91.

See "Configuring Array Policy Modules" on page 84.

About enabling or disabling controllers with shared disk groups

Prior to release 5.0, Veritas Volume Manager (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.

Multi-controller ALUA support

Multi-controller ALUA support enables:

- ALUA arrays with multiple storage controllers. DMP already supported storage arrays conforming to the ALUA standard, but the support was based on the traditional dual storage controller model.
- User-friendly CLI outputs which displays ALUA Asymmetric Access State (AAS) instead of legacy PRIMARY or SECONDARY states in the PATH-TYPE[M] column. For ALUA arrays, the DMP management interface displays the following ALUA states like:
  - Active/Optimized
  - Active/Non-optimized
  - Standby
  - Unavailable
  - TransitionInProgress
  - Offline

Note: The default value of the `dmp_display_alua_states` tunable is **on**. You can change the display mode to show legacy PRIMARY or SECONDARY path type by turning off the `dmp_display_alua_states` tunable.

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 9.

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 for new disk arrays. The Device Discovery uses a facility called the Device Discovery Layer (DDL).

The DDL enables you to add support for new disk arrays without the need for a reboot.

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.

VxVM supports the disk partitioning scheme provided by the operating system. The syntax of a device name is `hdx[N]` or `sdx[N]`, where *x* is a letter that indicates the order of EIDE (`hd`) or SCSI (`sd`) disks seen by the operating system, and *N* is an optional partition number in the range 1 through 15. An example of a device name is `sda7`, which references partition 7 on the first SCSI disk. If the partition number is omitted, the device name indicates the entire disk.

Devices that are specific to hardware from certain vendors may have different path names. For example, the COMPAQ SMART and SMARTII controllers use device names of the form `/dev/ida/cXdXpX` and `/dev/cciss/cXdXpX`.

Dynamic Multi-Pathing (DMP) uses the device name to create metadevices in the `/dev/vx/[r]dmp` directories. 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 9.

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

See "Disk device naming in DMP" on page 18.

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 90.

Device names can use one of the following naming schemes:

- operating system-based naming.
  See "About operating system-based naming" on page 18.
- enclosure-based naming.
  See "About enclosure-based naming" on page 19.

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 device naming scheme if required.

See "Changing the disk device naming scheme" on page 104.

About operating system-based naming

In the OS-based naming scheme, all disk devices are named using the `hdx[N]` format or the `sdx[N]` format, where *x* is a letter that indicates the order of EIDE (`hd`) or SCSI (`sd`) disks seen by the operating system, and *N* is an optional partition number in the range 1 through 15.

DMP assigns the name of the DMP meta-device (disk access name) from the multiple paths to the disk. DMP sorts the names alphabetically, and selects the first name. For example, sdc rather than sdd.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.

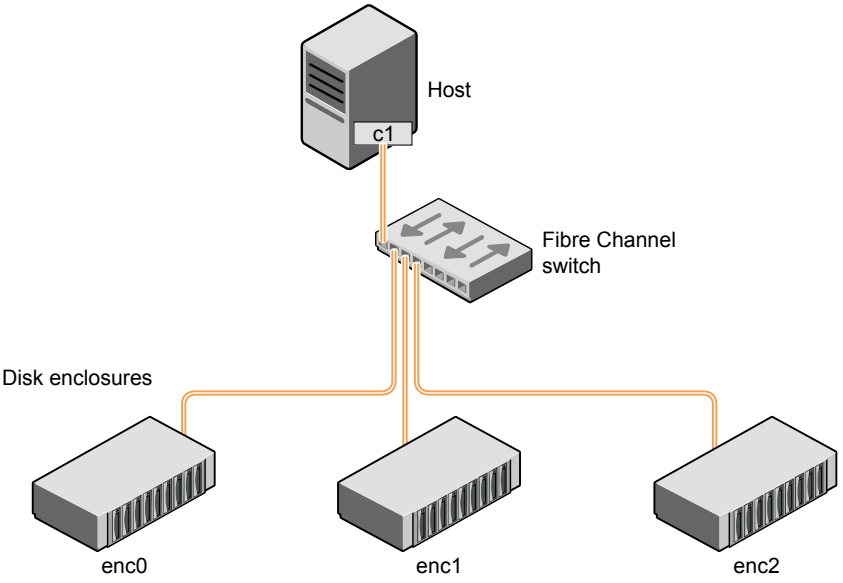
See "Changing the disk device naming scheme" on page 104.

About enclosure-based naming

Enclosure-based naming provides an alternative to operating system-based device 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 DMP 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 through 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 lets you quickly determine where a disk is physically located in a large SAN configuration.

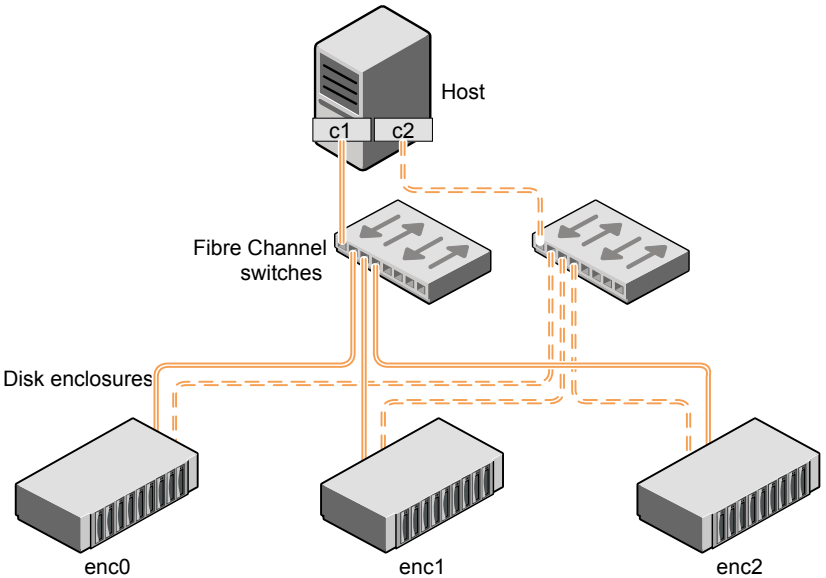
In most disk arrays, you can use hardware-based storage management to represent several physical disks as one LUN to the operating system. In such cases, VxVM also sees a single logical disk device rather than its component disks. For this reason, when reference is made to a disk within an enclosure,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

Another important benefit of enclosure-based naming is that it enables VxVM to avoid placing redundant copies of data in the same enclosure. This is a good thing to avoid as each enclosure can be considered to be a separate fault domain. For example, if a mirrored volume were configured only on the disks in enclosure `enc1`, the failure of the cable between the switch and the enclosure would make the entire volume unavailable.

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

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 (`c1` and `c2`), 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 `sdf` and `sdm`.

See "Disk device naming in DMP" on page 18.

See "Changing the disk device naming scheme" on page 104.

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

Summary of 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 the `vxdmpadm`(1M) manual page.

- Disks in the `DISKS` category (JBOD disks) are named using the `Disk_#` format.
- A disk partition is indicated by appending *`s#`* to the name, where `#` is the partition number. For example, `Disk_0s5` and `Disk_0s6` indicate the extended partitions that are used for the private and public regions of the sliced disk `Disk_0`. `ACME_0s5` indicates the extended partition for the simple disk, `ACME_0`. For CDS disks, partition 3 is used for both the private and public regions.
- Disks in the `OTHER_DISKS` category (disks that are not multi-pathed by DMP) are named using the `hdx[N]` format or the `sdx[N]` format.
- Encapsulated root disks always use the `hdx[N]` format or the `sdx[N]` format.

By default, enclosure-based names are persistent, so they do not change after a 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 "Disk categories" on page 90.

See "Enclosure based naming with the Array Volume Identifier (AVID) attribute" on page 22.

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

By default, Dynamic Multi-Pathing (DMP) assigns enclosure-based names to DMP metadevices 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 enclosure-based naming (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 third party drivers.

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 Cluster Volume Manager (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 Dynamic Multi-Pathing (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 volume groups to DMP devices. If the LVM volume groups are in use, then perform the steps to turn off the volume groups and migrate the volume groups 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 multi-pathed with Third-party drivers (TPD)	If a disk is already multi-pathed with a third-party driver (TPD), DMP does not manage the devices unless you remove TPD support. After removing TPD support, turn on the `dmp_native_support` tunable to migrate the devices. If LVM volume groups are constructed over TPD devices, then perform the steps to migrate the LVM volume groups onto DMP devices. See "Migrating LVM volume groups to DMP" on page 26.

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 that volume group. 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.

In case the volume cannot be disabled, such as a volume used by `/`, `/var` or `/home` file system, then a reboot should bring the volume groups onto the DMP device.

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
```

Displaying the native multi-pathing configuration

When DMP is enabled for native devices, the dmp_native_support tunable is set to 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.
  You can use the disks for native multi-pathing if you remove them from VxVM use.
- 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

Tunable                Current Value  Default Value
------------------     -------------  -------------
dmp_native_support     on              off
```

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

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
Linux	EMC PowerPath	See "Migrating to DMP from EMC PowerPath" on page 27.
Linux	Hitachi Data Link Manager (HDLM)	See "Migrating to DMP from Hitachi Data Link Manager (HDLM)" on page 28.
Linux	Linux Device Mapper Multipath	See "Migrating to DMP from Linux Device Mapper Multipath" on page 29.

Migrating to DMP from EMC PowerPath

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

Make sure that all paths belonging to the migrating PowerPath devices are in healthy state during the migration.

Plan for application downtime for the following procedure.

To remove devices from EMC PowerPath control and enable DMP

1. 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.

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

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

   ```
   # lvchange -a n lvpath
   ```

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

   ```
   # vxdisk rm emcpowerXXXX
   ```

   Where *emcpowerXXXX* is the name of the EMC PowerPath device.

5. Take the device out of PowerPath control:

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

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

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

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

```
# vxdisk scandisks
```

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

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

The above command also enables DMP support for LVM root.

A reboot is required for root support.

9 Mount the file systems.

10 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 application and system downtime for the following procedure.

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.

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

3 Stop the LVM volume groups that use the HDLM device.

```
# lvchange -a n lvpath
```

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 support for LVM root.

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 Linux Device Mapper Multipath

This procedure describes removing devices from Linux Device Mapper Multipath 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 remove devices from Device Mapper Multipath control and enable DMP

1 Stop the applications that use Device Mapper Multipath devices.

2 Unmount all the file systems that use Device Mapper Multipath devices.

3 Disable all the volumes on Device Mapper Multipath devices.

```
# lvchange -a n lvname
```

4 Update the `/etc/multipath.conf` file to blacklist all device mapper devices. This step disables multi-pathing for all devices.

```
# Blacklist all devices by default.
blacklist {
        devnode "*"
}
```

5 Restart `multipathd` to remove all `/dev/dm-*` and `/dev/mpath/*` device entries.

For systemd environments with supported Linux distributions:

```
# systemctl restart multipathd
```

For other supported Linux distributions:

```
# service multipathd restart
```

6 Stop the `multipathd` daemon.

For systemd environments with supported Linux distributions:

```
# systemctl stop multipathd
```

For other supported Linux distributions:

```
# service multipathd stop
```

7 Prevent `multipathd` from starting after reboot.

For systemd environments with supported Linux distributions:

```
# systemctl disable multipathd
```

For other supported Linux distributions:

```
# chkconfig multipathd off
```

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

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

The above command also enables DMP support for LVM root.

This step can take some time depending upon the number of LVs.

9 Mount the file systems.

10 Restart the applications.

Using Dynamic Multi-Pathing (DMP) devices with Oracle Automatic Storage Management (ASM)

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

- See "Enabling Dynamic Multi-Pathing (DMP) devices for use with Oracle Automatic Storage Management (ASM)" on page 31.
- See "Removing Dynamic Multi-Pathing (DMP) devices from the listing of Oracle Automatic Storage Management (ASM) disks" on page 32.

- See "Migrating Oracle Automatic Storage Management (ASM) disk groups on operating system devices to Dynamic Multi-Pathing (DMP) devices" on page 32.

Enabling Dynamic Multi-Pathing (DMP) devices for use with Oracle Automatic Storage Management (ASM)

Enable DMP support for Oracle Automatic Storage Management (ASM) to make DMP devices visible to ASM as available disks. DMP support for ASM is available for char devices (`/dev/vx/rdmp/*`).

To make DMP devices visible to ASM

1. From ASM, make sure ASM_DISKSTRING is set to the correct value:

   ```
   /dev/vx/rdmp/*
   ```

   For example:

   ```
   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.

   ```
   # vxdmpraw enable username groupname mode [devicename ...]
   ```

   where *username* represents the ASM user running the ASM instance, *groupname* represents the UNIX/Linux groupname of the specified user-id, and *mode* represents the permissions to set on the device. 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.

   For example:

   ```
   # vxdmpraw enable oracle dba 765 eva4k6k0_1
   ```

   ASM support is enabled. The access permissions for the DMP device are set to the permissions specified by *mode*. The changes are persistent across reboots.

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

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

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

4 From ASM, increase the Oracle heartbeat wait time from the default value of 15 seconds. To prevent the Oracle application from marking the disk as offline during the DMP failover, increase the default value for `_asm_hbeatiowait`.

- For example, to set the value to 360 seconds:

```
SQL> alter system set "_asm_hbeatiowait"=360 scope=spfile sid='*';
```

- Restart the ASM instance for the new parameter to take effect.

Removing Dynamic Multi-Pathing (DMP) devices from the listing of Oracle Automatic Storage Management (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.

```
# vxdmpraw disable diskname
```

For example:

```
# vxdmpraw disable eva4k6k0_1
```

Migrating Oracle Automatic Storage Management (ASM) disk groups on operating system devices to Dynamic Multi-Pathing (DMP) devices

When an existing ASM disk group uses operating system native devices as disks, you can migrate these devices to 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.

Plan for system downtime for the following procedure.

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. Stop the applications and shut down the database.
2. From ASM, identify the ASM disk group that you want to migrate, and identify the disks under its control.
3. From ASM, dismount the ASM disk group.
4. If the devices are controlled by other multi-pathing drivers, migrate the devices to DMP control. Perform these steps as root user.

   Migrate from PowerPath or Device Mapper Multipath.

   See "About setting up DMP to manage native devices" on page 24.
5. As root user, use the `raw` command to remove the raw devices that were created for the particular OS devices.
6. As root user, enable DMP support for the ASM disk group identified in step 2.

   ```
   # vxdmpraw enable username groupname mode [devicename ...]
   ```

   where *username* represents the ASM user running the ASM instance, *groupname* represents the UNIX/Linux groupname of the specified user-id, and *mode* represents the permissions to set on the device. 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.
7. From ASM, set ASM_DISKSTRING as appropriate. The preferred setting is `/dev/vx/rdmp/*`
8. From ASM, confirm that the devices are available to ASM.
9. 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 path , header_status from v$asm_disk where
header_status='MEMBER';

NAME         PATH             HEADER_STATUS
-----------------------------------------
ASM_DG1_0000 /dev/vx/rdmp/sda  MEMBER
ASM_DG1_0001 /dev/vx/rdmp/sdc  MEMBER
ASM_DG1_0002 /dev/vx/rdmp/sdd  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.

See "About setting up DMP to manage native devices" on page 24.

4 As root user, enable DMP support for the ASM disk group identified in step 2, 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=sdd
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:

```
# vxdmpraw enable oracle dba 660 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:

  ```
  # vxdmpraw enable oracle dba 660
  ```

5 From ASM, set ASM_DISKSTRING.

```
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 path , header_status from v$asm_disk where
header_status='MEMBER';


NAME   PATH                              HEADER_STATUS
-------------------------------------------------
       /dev/vx/rdmp/emc_clariion0_243  MEMBER
       /dev/vx/rdmp/eva4k6k4k0_0       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 path , header_status from v$asm_disk where
header_status='MEMBER';

NAME          PATH                              HEADER_STATUS
-------------------------------------------------------------
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.

Use the `vxdisk list` command to identify these types of disks.

- Disks that are not in use by VxVM
  The output of `vxdisk list` shows these disks with the Type auto:none and the Status as online invalid.

The example shows available disks.

```
# vxdisk list

DEVICE               TYPE      DISK  GROUP    STATUS
. . .
tagmastore-usp0_0035 auto:none -     -        online invalid
tagmastore-usp0_0036 auto:none -     -        online invalid
```

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

Use the complete path name for the DMP device.

```
# pvcreate /dev/vx/dmp/tagmastore-usp0_0035
Physical volume "/dev/vx/dmp/tagmastore-usp0_0035" successfully
created
#
# vgcreate /dev/newvg /dev/vx/dmp/tagmastore-usp0_0035
  Volume group "newvg" successfully created
# vgdisplay -v newvg |grep Name
    Using volume group(s) on command line
    Finding volume group "newvg"
  VG Name               newvg
    PV Name               /dev/vx/dmp/tagmastore-usp0_0035s3
```

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

Use the complete path name for the DMP device.

```
# pvcreate /dev/vx/dmp/tagmastore-usp0_0036
Physical volume "/dev/vx/dmp/tagmastore-usp0_0036"
successfully created

# vgextend newvg /dev/vx/dmp/tagmastore-usp0_0036
  Volume group "newvg" successfully extended

# vgdisplay -v newvg |grep Name
    Using volume group(s) on command line
    Finding volume group "newvg"
  VG Name               newvg
  PV Name               /dev/vx/dmp/tagmastore-usp0_0035s3
  PV Name               /dev/vx/dmp/tagmastore-usp0_0036s3
```

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

```
# vxdisk scandisks
```

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

```
# vxdisk list

. . .
tagmastore-usp0_0035 auto:LVM   -         -    LVM
tagmastore-usp0_0036 auto:LVM   -         -    LVM
```

6 For all of the LVM volume entries, add '_netdev' to the mount options in /etc/fstab. This option ensures that these volumes are enabled after DMP devices are discovered.

Removing DMP support for native devices

The dmp_native_support tunable is persistent across reboots and product upgrades. You can disable native support for an individual device 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.

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
```

Chapter 3

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 through a Host Bus Adapter (HBA) controller or the array port of a storage processor so that you can perform administrative operations. This feature can be used when you perform maintenance on HBA controllers on the host, or array ports that are attached to disk arrays supported by DMP. I/O operations to the HBA controller or the 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, when you disable 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 secondary paths and I/O will continue on them.

After the administrative operation is over, use the `vxdmpadm` command to re-enable the paths through the HBA controllers or array ports.

See “Disabling I/O for paths, controllers, array ports, or DMP nodes” on page 74.

See "Enabling I/O for paths, controllers, array ports, or DMP nodes" on page 76.

You can also perform certain reconfiguration operations dynamically online.

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 the vxdmpadm utility" on page 44.

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 physical disk having multiple physical paths through the system's HBA controllers. In Dynamic Multi-Pathing (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
sda          sda        mydg01     mydg      ENABLED
sdi          sdi        mydg01     mydg      ENABLED
sdb          sdb        mydg02     mydg      ENABLED
sdj          sdj        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 the device `sdl`, use the following command:

```
# vxdisk list sdl
```

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

```
Device:    sdl
devicetag: sdl
type:      sliced
hostid:    sys1
.
.
.
Multipathing information:
numpaths:   2
sdl    state=enabled     type=primary
sdp    state=disabled    type=secondary
```

The `numpaths` line shows that there are 2 paths to the device. The next two lines in the "Multipathing information" section of the output 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_431`, use the following command:

```
# # vxdmpadm getsubpaths dmpnodename=emc_clariion0_431
```

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

```
NAME   STATE[A]    PATH-TYPE[M]          CTLR-NAME  ENCLR-TYPE    ENCLR-NAME    ATTRS PRIORITY
=============================================================================================
sdac   ENABLED     Active/Non-Optimized  c6         EMC_CLARiiON  emc_clariion0  -       -
sdam   ENABLED(A)  Active/Optimized(P)   c6         EMC_CLARiiON  emc_clariion0  -       -
sdi    ENABLED     Active/Non-Optimized  c1         EMC_CLARiiON  emc_clariion0  -       -
sds    ENABLED(A)  Active/Optimized(P)   c1         EMC_CLARiiON  emc_clariion0  -       -
```

Setting customized names for DMP nodes

The Dynamic Multi-Pathing (DMP) node name is the metadevice name that represents the multiple paths to a disk. The Device Discovery Layer (DDL) generates the DMP node name from the device name according to the Dynamic Multi-Pathing (DMP) naming scheme.

See "Disk device naming in DMP" on page 18.

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 specify a custom name for an enclosure

- Use the following command:

```
# vxdmpadm setattr enclosure enc_name name=custom_name
```

To assign DMP nodes from a file

1. To obtain a file populated with the names of the devices in your configuration, use the following command:

```
# vxddladm -l assign names  > filename
```

   The sample file shows the format required and serves as a template to specify your customized names.

   You can also use the script `vxgetdmpnames` to get a sample file populated from the devices in your configuration.

2. Modify the file as required. Be sure to maintain the correct format in the file.
3. To assign the names, specify the name and path of the file to the following command:

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

To clear custom names

- To clear the names, and use the default operating system-based naming or enclosure-based naming, use the following command:

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

Administering DMP using the vxdmpadm utility

The `vxdmpadm` utility is a command-line administrative interface to Dynamic Multi-Pathing (DMP).

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

- Retrieve the name of the DMP device corresponding to a particular path.
  See "Retrieving information about a DMP node" on page 46.
- Display consolidated information about the DMP nodes.
  See "Displaying consolidated information about the DMP nodes" on page 47.
- Display the members of a LUN group.
  See "Displaying the members of a LUN group" on page 48.

- List all paths under a DMP device node, HBA controller, enclosure, or array port.
  See "Displaying paths controlled by a DMP node, controller, enclosure, or array port" on page 49.
- Display information about the HBA controllers on the host.
  See "Displaying information about controllers" on page 51.
- Display information about enclosures.
  See "Displaying information about enclosures" on page 53.
- Display information about array ports that are connected to the storage processors of enclosures.
  See "Displaying information about array ports" on page 53.
- Display asymmetric access state for ALUA arrays.
  See "User-friendly CLI outputs for ALUA arrays " on page 54.
- Display information about devices that are controlled by third-party multi-pathing drivers.
  See "Displaying information about devices controlled by third-party drivers" on page 55.
- Display extended devices attributes.
  See "Displaying extended device attributes" on page 56.
- See "Suppressing or including devices from VxVM control" on page 58.
  Suppress or include devices from DMP control.
- Gather I/O statistics for a DMP node, enclosure, path, or controller.
  See "Gathering and displaying I/O statistics" on page 59.
- Configure the attributes of the paths to an enclosure.
  See "Setting the attributes of the paths to an enclosure" on page 65.
- Display the redundancy level of a device or enclosure.
  See "Displaying the redundancy level of a device or enclosure" on page 66.
- Specify the minimum number of active paths.
  See "Specifying the minimum number of active paths" on page 67.
- Display or set the I/O policy that is used for the paths to an enclosure.
  See "Specifying the I/O policy" on page 68.
- Enable or disable I/O for a path, HBA controller or array port on the system.
  See "Disabling I/O for paths, controllers, array ports, or DMP nodes" on page 74.
- Rename an enclosure.
  See "Renaming an enclosure" on page 77.

- Configure how DMP responds to I/O request failures.
  See "Configuring the response to I/O failures" on page 77.
- Configure the I/O throttling mechanism.
  See "Configuring the I/O throttling mechanism" on page 79.
- Control the operation of the DMP path restoration thread.
  See "Configuring DMP path restoration policies" on page 82.
- Configure array policy modules.
  See "Configuring Array Policy Modules" on page 84.
- Get or set the values of various tunables used by DMP.
  See "DMP tunable parameters" on page 147.

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

Retrieving information about a DMP node

The following command displays the Dynamic Multi-Pathing (DMP) node that controls a particular physical path:

```
# vxdmpadm getdmpnode nodename=pathname
```

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

The device directory is the `/dev` directory.

The command displays output similar to the following example output.

```
# vxdmpadm getdmpnode nodename=sdbc

NAME               STATE    ENCLR-TYPE    PATHS ENBL DSBL ENCLR-NAME
=============================================================================
emc_clariion0_89 ENABLED EMC_CLARiiON 6     6    0    emc_clariion0
```

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

```
# vxdmpadm -v getdmpnode nodename=sdbc

NAME              STATE  ENCLR-TYPE  PATHS ENBL DSBL ENCLR-NAME SERIAL-NO ARRAY_VOL_ID
=============================================================================================
emc_clariion0_89 ENABLED EMC_CLARiiON 6   6    0   emc_clariion0 600601601 893
```

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

```
# vxdmpadm getdmpnode enclosure=emc_clariion0
```

```
NAME                STATE    ENCLR-TYPE    PATHS  ENBL  DSBL  ENCLR-NAME
=============================================================================
emc_clariion0_429   ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_430   ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_431   ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_432   ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
```

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 Dynamic Multi-Pathing (DMP) node. The information includes the enclosure name, LUN serial number, port id information, device attributes, and so on.

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=enclosurename
```

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 DMP node.

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_158
state       = enabled
enclosure   = emc_clariion0
cab-sno     = CK200070400359
asl         = libvxCLARiiON.so
vid         = DGC
pid         = DISK
array-name  = EMC_CLARiiON
array-type  = CLR-A/PF
iopolicy    = MinimumQ
avid        = 158
lun-sno     = 600601601A141B001D4A32F92B49DE11
udid        = DGC%5FDISK%5FCK200070400359%5F600601601A141B001D4A32F92B49DE11
dev-attr    = lun
###path     = name state type transport ctlr hwpath aportID aportWWN attr
path        = sdck enabled(a) primary FC c2 c2 A5 50:06:01:61:41:e0:3b:33 -
path        = sdde enabled(a) primary FC c2 c2 A4 50:06:01:60:41:e0:3b:33 -
path        = sdcu enabled secondary FC c2 c2 B4 50:06:01:68:41:e0:3b:33 -
path        = sdbm enabled secondary FC c3 c3 B4 50:06:01:68:41:e0:3b:33 -
path        = sdbw enabled(a) primary FC c3 c3 A4 50:06:01:60:41:e0:3b:33 -
path        = sdbc enabled(a) primary FC c3 c3 A5 50:06:01:61:41:e0:3b:33 -
```

Displaying the members of a LUN group

The following command displays the Dynamic Multi-Pathing (DMP) nodes that are in the same LUN group as a specified DMP node:

```
# vxdmpadm getlungroup dmpnodename=dmpnode
```

For example:

```
# vxdmpadm getlungroup dmpnodename=sdq
```

```
NAME     STATE    ENCLR-TYPE  PATHS     ENBL    DSBL   ENCLR-NAME
============================================================
sdo      ENABLED  ACME        2         2       0       enc1
sdp      ENABLED  ACME        2         2       0       enc1
```

```
sdq       ENABLED  ACME         2          2        0        enc1
sdr       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 Dynamic Multi-Pathing (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 worldwide name (WWN).

To list all subpaths known to DMP:

```
# vxdmpadm getsubpaths
```

```
NAME    STATE[A]    PATH-TYPE[M]          DMPNODENAME        ENCLR-NAME    CTLR    ATTRS
=============================================================================
sdaf    ENABLED(A)  PRIMARY               ams_wms0_130       ams_wms0      c2        -
sdc     ENABLED     SECONDARY             ams_wms0_130       ams_wms0      c3        -
sdb     ENABLED(A)    -                   vm04_disk_24       disk          c0        -
sda     ENABLED(A)    -                   vm04_disk_25       disk          c0        -
sdaa    ENABLED     Active/Non-Optimized emc_clariion0_438 emc_clariion0 c1        -
sdak    ENABLED(A)  Active/Optimized(P)   emc_clariion0_438 emc_clariion0 c6        -
```

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/dmp` directory:

```
# vxdmpadm getsubpaths dmpnodename=sdby
```

```
NAME    STATE[A]    PATH-TYPE[M] CTLR-NAME  ENCLR-TYPE  ENCLR-NAME  ATTRS    PRIORITY
=============================================================================
sdbp    ENABLED       -          c1          EMC         emc0          -        -
sdbs    ENABLED       -          c1          EMC         emc0          -        -
sdbv    ENABLED       -          c6          EMC         emc0          -        -
sdby    ENABLED(A)    -          c6          EMC         emc0          -        -
```

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 68.

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=c1

NAME STATE[A]   PATH-TYPE[M]         DMPNODENAME        ENCLR-TYPE   ENCLR-NAME     ATTRS  PRIOR
=============================================================================================================
sdh  ENABLED    Active/Non-Optimized emc_clariion0_429  EMC_CLARiiON  emc_clariion0     -
sdr  ENABLED(A) Active/Optimized(P)  emc_clariion0_429  EMC_CLARiiON  emc_clariion0     -
sdm  ENABLED(A) Active/Optimized(P)  emc_clariion0_430  EMC_CLARiiON  emc_clariion0     -
sdw  ENABLED    Active/Non-Optimized emc_clariion0_430  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 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=A7

NAME STATE[A]   PATH-TYPE[M]         DMPNODENAME       ENCLR-NAME   CTLR ATTRS PRIORITY
=============================================================================================
sdal ENABLED(A) Active/Optimized     emc_clariion0_429 emc_clariion0 c6   -      -
sdr  ENABLED(A) Active/Optimized     emc_clariion0_429 emc_clariion0 c1   -      -
sdaq ENABLED    Active/Non-Optimized emc_clariion0_430 emc_clariion0 c6   -      -
sdw  ENABLED    Active/Non-Optimized emc_clariion0_430 emc_clariion0 c1   -      -
```

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

```
# vxdmpadm getsubpaths pwwn=50:06:01:67:3e:a0:75:95

NAME STATE[A]   PATH-TYPE[M]         CTLR-NAME ENCLR-TYPE   ENCLR-NAME     ATTRS PRIORITY
=============================================================================================
sdal ENABLED(A) Active/Optimized     c6        EMC_CLARiiON  emc_clariion0  -       -
sdr  ENABLED(A) Active/Optimized(P)  c1        EMC_CLARiiON  emc_clariion0  -       -
sdaq ENABLED    Active/Non-Optimized c6        EMC_CLARiiON  emc_clariion0  -       -
sdw  ENABLED    Active/Non-Optimized c1        EMC_CLARiiON  emc_clariion0  -       -
```

```
# vxdmpadm getsubpaths pwwn=20:00:00:E0:8B:06:5F:19
```

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
================================================================================
sdav         ENABLED(A) PRIMARY      emc_clariion0_1017 emc_clariion0 c3    -
sdbf         ENABLED    SECONDARY    emc_clariion0_1017 emc_clariion0 c3    -
sdau         ENABLED(A) PRIMARY      emc_clariion0_1018 emc_clariion0 c3    -
sdbe         ENABLED    SECONDARY    emc_clariion0_1018 emc_clariion0 c3    -
```

To list all subpaths of a controller on an enclosure:

```
# vxdmpadm getsubpaths enclosure=emc_clariion0
```

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]
```

See "Setting customized names for DMP nodes" on page 43.

Displaying information about controllers

The following Dynamic Multi-Pathing (DMP) command lists attributes of all HBA controllers on the system:

```
# vxdmpadm listctlr all

CTLR-NAME  ENCLR-TYPE     STATE    ENCLR-NAME      PATH_COUNT
==============================================================
c1         OTHER          ENABLED  other0          3
c2         X1             ENABLED  jbod0           10
```

```
c3          ACME            ENABLED  enc0            24
c4          ACME            ENABLED  enc0            24
```

This output shows that the controller `c1` 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=emc0
```

or

```
# vxdmpadm listctlr type=EMC
```

```
# vxdmpadm listctlr type=EMC
CTLR_NAME   ENCLR_TYPE    STATE      ENCLR_NAME  PATH_COUNT
============================================================
c1          EMC           ENABLED    emc0             6
c6          EMC           ENABLED    emc0             6
```

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 occurrence of events. The WWN shown as ‘Controller ID’ maps to the WWN of the HBA port associated with the host controller.

```
# vxdmpadm getctlr c5
```

```
LNAME      PNAME VENDOR  CTLR-ID
================================================
c5         c5    qlogic      20:07:00:a0:b8:17:e1:37
```

Displaying information about enclosures

Dynamic Multi-Pathing (DMP) can display the attributes of the enclosures, including the enclosure type, enclosure serial number, status, array type, number of LUNs, and the firmware version, if available.

To display the attributes of a specified enclosure, use the following DMP command:

```
# vxdmpadm listenclosure emc0
ENCLR_NAME ENCLR_TYPE ENCLR_SNO            STATUS    ARRAY_TYPE LUN_COUNT FIRMWARE
==============================================================================
emc0       EMC        000292601383         CONNECTED A/A        30        5875
```

To display the attrtibutes for all enclosures in a system, use the following DMP command:

```
# vxdmpadm listenclosure all
```

```
ENCLR_NAME      ENCLR_TYPE     ENCLR_SNO     STATUS    ARRAY_TYPE LUN_COUNT FIRMWARE
==============================================================================
Disk            Disk           DISKS         CONNECTED Disk       6         -
emc0            EMC            000292601383  CONNECTED A/A        1         5875
hitachi_usp-vm0 Hitachi_USP-VM 25847         CONNECTED A/A        1         6008
emc_clariion0   EMC_CLARiiON   CK20007040035 CONNECTED CLR-A/PF   2         0324
```

Displaying information about array ports

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

To display the attributes of an array port that is accessible through 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 through DMP node `sdg`:

```
# vxdmpadm getportids dmpnodename=sdg

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

User-friendly CLI outputs for ALUA arrays

DMP supports storage arrays using ALUA standard. From Veritas InfoScale 7.1 onwards, DMP supports multi-controller (more than 2 controllers) ALUA compliant arrays.

For ALUA arrays, the `dmp_display_alua_states` tunable parameter displays the asymmetric access state of the Logical Unit (LUN) instead of PRIMARY or SECONDARY in the PATH-TYPE[M] column.

Note: The default tunable value is **on**.

To view asymmetric access states of an ALUA LUN, enter:

```
# vxdmpadm getsubpaths dmpnodename=dmpnode_name
```

Typical output is as follows:

```
# vxdmpadm getsubpaths dmpnodename=emc_clariion0_786

NAME     STATE[A]    PATH-TYPE[M]         CTLR-NAME ENCLR-TYPE    ENCLR-NAME    ATTRS
=============================================================================================
hdisk40  ENABLED     Active/Non-Optimized fscsi0    EMC_CLARiiON  emc_clariion0     -
hdisk58  ENABLED     Active/Non-Optimized fscsi1    EMC_CLARiiON  emc_clariion0     -
hdisk67  ENABLED(A)  Active/Optimized(P)  fscsi1    EMC_CLARiiON  emc_clariion0     -
hdisk77  ENABLED(A)  Active/Optimized(P)  fscsi0    EMC_CLARiiON  emc_clariion0     -
```

Note: In the output, (P) signifies that the path is connected to the target port group marked as preferred by the device server.

All VxVM/DMP outputs which earlier displayed PRIMARY or SECONDARY in the PATH-TYPE[M] column will now display the asymmetric access state.

If you want to go back to the previous version of the CLI output which displays PRIMARY or SECONDARY in the PATH-TYPE[M] column, enter the following command to disable the `dmp_display_alua_states` tunable parameter:

```
# vxdmpadm settune dmp_display_alua_states=off
```

The tunable value changes immediately.

Displaying information about devices controlled by third-party drivers

The third-party driver (TPD) coexistence feature allows I/O that is controlled by third-party multi-pathing drivers to bypass Dynamic Multi-Pathing (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
```

See "Changing device naming for enclosures controlled by third-party drivers" on page 107.

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
emcpowerp    auto:cdsdisk    -            -            online
emcpowerq    auto:cdsdisk    -            -            online
emcpowerr    auto:cdsdisk    -            -            online
emcpowers    auto:cdsdisk    -            -            online
emcpowert    auto:cdsdisk    -            -            online
```

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

```
# vxdmpadm getsubpaths tpdnodename=emcpowerp

NAME         TPDNODENAME   PATH-TYPE[-] DMPNODENAME   ENCLR-TYPE    ENCLR-NAME
=============================================================================
sdt          emcpowerp        -        emcpowerp     PP_EMC_CLARiiON  pp_emc_clariion0
sdo          emcpowerp        -        emcpowerp     PP_EMC_CLARiiON  pp_emc_clariion0
sdj          emcpowerp        -        emcpowerp     PP_EMC_CLARiiON  pp_emc_clariion0
sde          emcpowerp        -        emcpowerp     PP_EMC_CLARiiON  pp_emc_clariion0
```

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

```
# vxdmpadm gettpdnode nodename=sdt
NAME                 STATE     PATHS     ENCLR-TYPE    ENCLR-NAME
```

```
==============================================================================
emcpowerp            ENABLED      4        PP_EMC_CLARiiON  pp_emc_clariion0
```

Displaying extended device attributes

Device Discovery Layer (DDL) extended attributes are attributes or flags corresponding to a Veritas Volume Manager (VxVM) or Dynamic Multi-Pathing (DMP) LUN or disk and that are discovered by DDL. These attributes identify a LUN to a specific hardware category.

Table 3-1 describes the list of categories.

Table 3-1 Categories for extended attributes

Category	Description
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 Drive)
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 extended attributes. DDL discovers the extended attributes during device discovery from the Array Support Library (ASL). If Veritas Operations Manager (VOM) is present, DDL can also obtain extended attributes from the VOM Management Server for hosts that are configured as managed hosts.

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

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 (HCL).

Suppressing or including devices from VxVM control

The `vxdmpadm exclude` command suppresses devices from Veritas Volume Manager (VxVM) based on the criteria that you specify. When a device is suppressed, Dynamic Multi-Pathing (DMP) does not claim the device so that the device is not available for VxVM to use. You can add the devices back into VxVM control with 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 encapsulated under VxVM.

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

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

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

where:

all	all devices
product=*VID:PID*	all devices with the specified VID:PID
ctlr=*ctlrname*	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, port, or controller.

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.

To enable the gathering of statistics, enter this command:

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

The `memory` attribute limits 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 reset the I/O counters to zero, use this command:

```
# vxdmpadm iostat reset
```

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

```
# vxdmpadm iostat show {filter} [interval=seconds [count=N]]
```

The above command displays I/O statistics for the devices specified by the *filter*. The *filter* is one of the following:

- `all`
- `ctlr=`*`ctlr-name`*
- `dmpnodename=`*`dmp-node`*
- `enclosure=`*`enclr-name`* `[portid=`*`array-portid`* `] [ctlr=`*`ctlr-name`*`]`
- `pathname=`*`path-name`*
- `pwwn=`*`array-port-wwn`* `[ctlr=`*`ctlr-name`*`]`

Use the `interval` and `count` attributes 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.

DMP also provides a *groupby* option to display cumulative I/O statistics, aggregated by the specified criteria.

See "Displaying cumulative I/O statistics" on page 60.

To disable the gathering of statistics, enter this command:

```
# vxdmpadm iostat stop
```

Displaying cumulative I/O statistics

The `vxdmpadm iostat` command provides the ability to analyze the I/O load distribution across various I/O channels or parts of I/O channels. Select the appropriate *filter* to display the I/O statistics for the DMP node, controller, array enclosure, path, port, or virtual machine. Then, use the *groupby* clause to display cumulative statistics according to the criteria that you want to analyze. If the `groupby` clause is not specified, then the statistics are displayed per path.

When you combine the *filter* and the *groupby* clause, you can analyze the I/O load for the required use case scenario. For example:

- To compare I/O load across HBAs, enclosures, or array ports, use the *groupby* clause with the specified attribute.
- To analyze I/O load across a given I/O channel (HBA to array port link), use *filter* by HBA and PWWN or enclosure and array port.
- To analyze I/O load distribution across links to an HBA, use *filter* by HBA and *groupby* array port.

Use the following format of the `iostat` command to analyze the I/O loads:

```
# vxdmpadm [-u unit] iostat show [groupby=criteria] {filter}  \
  [interval=seconds [count=N]]
```

The above command displays I/O statistics for the devices specified by the *filter*. The *filter* is one of the following:

- `all`
- `ctlr=ctlr-name`
- `dmpnodename=dmp-node`
- `enclosure=enclr-name [portid=array-portid ] [ctlr=ctlr-name]`
- `pathname=path-name`
- `pwwn=array-port-wwn[ctlr=ctlr-name]`

You can aggregate the statistics by the following `groupby` criteria:

- `arrayport`
- `ctlr`

- dmpnode
- enclosure

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. You can specify the units in which the statistics data is displayed. The -u option accepts the following options:

h or H	Displays throughput in the highest possible unit.
k	Displays throughput in kilobytes.
m	Displays throughput in megabytes.
g	Displays throughput in gigabytes.
bytes| b	Displays throughput in exact number of bytes.
us	Displays average read/write time in microseconds.

To group by DMP node:

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

To group by controller:

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

For example:

```
# vxdmpadm iostat show groupby=ctlr ctlr=c5

             OPERATIONS        BLOCKS       AVG TIME(ms)
CTLRNAME  READS    WRITES  READS  WRITES  READS    WRITES
c5        224       14     54      7       4.20     11.10
```

To group by arrayport:

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

For example:

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

```
            OPERATIONS        BYTES       AVG TIME(ms)
PORTNAME  READS   WRITES  READS  WRITES  READS   WRITES
1A        743     1538    11m    24m     17.13   8.61
```

To group by enclosure:

```
# vxdmpadm [-u unit] iostat show groupby=enclosure [ all \
| enclosure=enclr ]
```

For example:

```
# vxdmpadm -u h iostat show groupby=enclosure enclosure=EMC_CLARiiON0

OPERATIONS       BLOCKS       AVG TIME(ms)
ENCLOSURENAME   READS  WRITES  READS  WRITES  READS   WRITES
EMC_CLARiiON0   743    1538    11392k  24176k 17.13   8.61
```

You can also filter out entities for which all data entries are zero. This option is especially useful in a cluster environment that 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 [-u unit] -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=emc_clariion0_893

cpu usage = 9852us    per cpu memory = 266240b
             OPERATIONS      BLOCKS          AVG TIME(ms)
PATHNAME  READS  WRITES   READS   WRITES   READS    WRITES
sdbc      32     0        258     0         0.04      0.00
sdbw      27     0        216     0         0.03      0.00
sdck      8      0        57      0         0.04      0.00
sdde      11     0        81      0         0.15      0.00
```

To display average read/write times in microseconds.

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

cpu usage = 9865us    per cpu memory = 266240b
              OPERATIONS       BLOCKS           AVG TIME(us)
PATHNAME   READS    WRITES  READS    WRITES   READS    WRITES
sdck       8        0       57       0        43.04      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 Dynamic Multi-Pathing (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=emc_clariion0_352

cpu usage = 338us    per cpu memory = 102400b
                        QUEUED I/Os     PENDING I/Os
DMPNODENAME          READS    WRITES
emc_clariion0_352    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=sdo

cpu usage = 637us    per cpu memory = 102400b
                  ERROR I/Os
PATHNAME      READS    WRITES
sdo           0         0
```

Examples of using the vxdmpadm iostat command

Dynamic Multi-Pathing (DMP) enables you to gather and display I/O statistics with the `vxdmpadm iostat` command. This section provides 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 the kilobytes read and written, on all paths.

```
# vxdmpadm -u k iostat show all
                    cpu usage = 7952us    per cpu memory = 8192b
             OPERATIONS           BYTES         AVG TIME(ms)
PATHNAME READS     WRITES    READS     WRITES   READS    WRITES
sdf          87          0    44544k          0    0.00      0.00
sdk           0          0        0          0    0.00      0.00
sdg          87          0    44544k          0    0.00      0.00
sdl           0          0        0          0    0.00      0.00
sdh          87          0    44544k          0    0.00      0.00
sdm           0          0        0          0    0.00      0.00
sdi          87          0    44544k          0    0.00      0.00
sdn           0          0        0          0    0.00      0.00
sdj          87          0    44544k          0    0.00      0.00
sdo           0          0        0          0    0.00      0.00
sdj          87          0    44544k          0    0.00      0.00
sdp           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 -u k iostat show pathname=sdk
                    cpu usage = 8132us    per cpu memory = 4096b
               OPERATIONS            BYTES             AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES     READS    WRITES
sdk             0          0          0          0      0.00      0.00

# vxdmpadm -u k iostat show dmpnodename=sdf
                    cpu usage = 8501us    per cpu memory = 4096b
               OPERATIONS            BYTES             AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES     READS    WRITES
sdf          1088          0     557056k          0      0.00      0.00

# vxdmpadm -u k iostat show enclosure=Disk
                   cpu usage = 8626us    per cpu memory = 4096b
```

```
                 OPERATIONS              BYTES              AVG TIME(ms)
PATHNAME     READS      WRITES       READS      WRITES      READS    WRITES
sdf           1088           0     557056k           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=sdk interval=2 count=2
                    cpu usage = 9621us    per cpu memory = 266240b
                 OPERATIONS            BLOCKS          AVG TIME(ms)
PATHNAME     READS     WRITES      READS     WRITES      READS     WRITES
sdk          0          0          0          0          0.00       0.00

sdk          0          0          0          0          0.00       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 across reboots or product upgrades.

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 sde 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 sdm pathtype=nomanual
```

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

```
# vxdmpadm setattr path sdk \
  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.

See "Specifying the I/O policy" on page 68.

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 sdk 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 sdm 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 sdn 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 sde 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 Dynamic Multi-Pathing (DMP) log file. Also, notification is sent to `vxnotify` clients.

The value set for minimum redundancy level is persistent across reboots and product upgrades. 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            512             4096
```

Specifying the I/O policy

You can use the `vxdmpadm setattr` command to change the Dynamic Multi-Pathing (DMP) 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).

Note: I/O policies are persistent across reboots of the system.

Table 3-2 describes the I/O policies that may be set.

Table 3-2 DMP I/O policies

Policy	Description

`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`

`adaptiveminq`	Similar to the `adaptive` policy, except that I/O is scheduled according to the length of the I/O queue on each path. The path with the shortest queue is assigned the highest priority.

Table 3-2 DMP I/O policies *(continued)*

Policy	Description

`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 partitionsize attribute to specify the size for the partition. 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 512 blocks (256k). Specifying a partition size of 0 is equivalent to the default partition size of 512 blocks (256k).
	
	The default value can be changed by adjusting the value of the `dmp_pathswitch_blks_shift` tunable parameter.
	
	See "DMP tunable parameters" on page 147.
	
	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.

Table 3-2 DMP I/O policies *(continued)*

Policy	Description

`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 65.
	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 or an ALUA 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 or an ALUA 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 or ALUA.

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    ATTR_NAME     DEFAULT CURRENT
============================================
HDS9500-ALUA0 use_all_paths no      yes
```

The `use_all_paths` attribute only applies to A/A-A arrays and ALUA 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 use Dynamic Multi-Pathing (DMP) to configure load balancing in a SAN environment where there are multiple primary paths to an Active/Passive device through several SAN switches.

As shown in this sample output from the `vxdisk list` command, the device `sdm` has eight primary paths:

```
# vxdisk list sdq

Device: sdq
    .
    .
    .
numpaths: 8
sdj state=enabled type=primary
sdk state=enabled type=primary
sdl state=enabled type=primary
```

```
sdm state=enabled type=primary
sdn state=enabled type=primary
sdo state=enabled type=primary
sdp state=enabled type=primary
sdq 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, use the `dd` command 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, `sdq`:

```
# vxdmpadm iostat show dmpnodename=sdq interval=5 count=2
    .
    .
    .
cpu usage = 11294us per cpu memory = 32768b
             OPERATIONS           KBYTES          AVG TIME(ms)
PATHNAME   READS    WRITES    READS     WRITES    READS       WRITES
sdj        0        0         0         0          0.00         0.00
sdk        0        0         0         0          0.00         0.00
sdl        0        0         0         0          0.00         0.00
sdm        0        0         0         0          0.00         0.00
sdn        0        0         0         0          0.00         0.00
sdo        0        0         0         0          0.00         0.00
sdp        0        0         0         0          0.00         0.00
sdq        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=sdq interval=5 count=2
    .
    .
    .
cpu usage = 14403us per cpu memory = 32768b
             OPERATIONS            KBYTES         AVG TIME(ms)
PATHNAME   READS    WRITES    READS     WRITES   READS      WRITES
sdj        2041     0         1021      0         0.39        0.00
sdk        1894     0         947       0         0.39        0.00
sdl        2008     0         1004      0         0.39        0.00
sdm        2054     0         1027      0         0.40        0.00
sdn        2171     0         1086      0         0.39        0.00
sdo        2095     0         1048      0         0.39        0.00
sdp        2073     0         1036      0         0.39        0.00
sdq        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, array ports, or DMP nodes

Disabling I/O through a path, HBA controller, array port, or Dynamic Multi-Pathing (DMP) node prevents DMP from issuing I/O requests through the specified path, or the paths that are connected to the specified controller, array port, or DMP node.

If the specified paths have pending I/Os, the `vxdmpadm disable` command waits until the I/Os are completed before disabling the paths.

DMP does not support the operation to disable I/O for the controllers that use Third-Party Drivers (TPD) for multi-pathing.

To disable I/O for one or more 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 one or more HBA controllers, use the following command:

```
# vxdmpadm [-c|-f] disable ctlr=ctlr_name1[,ctlr_name2,ctlr_nameN]
```

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 examples show how 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
```

To disable I/O for a particular path, specify both the controller and the portID, which represent the two ends of the fabric:

```
# vxdmpadm [-c|-f] disable ctlr=ctlr_name enclosure=enclr_name \
portid=array_port_ID
```

To disable I/O for a particular DMP node, specify the DMP node name.

```
# vxdmpadm [-c|-f] disable dmpnodename=dmpnode
```

You can use the `-c` option to check if there is only a single active path to the disk.

Use the **-f** option to disable the last path, irrespective of whether the device is in use or not.

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. The `disable` operation fails if it is issued to a controller that is connected to the swap device through a single path.

Enabling I/O for paths, controllers, array ports, or DMP nodes

Enabling a controller allows a previously disabled path, HBA controller, array port, or Dynamic Multi-Pathing (DMP) node to accept I/O again. This operation succeeds only if the path, controller, array port, or DMP node 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: This operation is supported for controllers that are used to access disk arrays on which cluster-shareable disk groups are configured.

DMP does not support the operation to enable I/O for the controllers that use Third-Party Drivers (TPD) for multi-pathing.

To enable I/O for one or more paths, use the following command:

```
# vxdmpadm enable path=path_name1[,path_name2,path_nameN]
```

To enable I/O for the paths connected to one or more HBA controllers, use the following command:

```
# vxdmpadm enable ctlr=ctlr_name1[,ctlr_name2,ctlr_nameN]
```

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 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
```

To enable I/O for a particular path, specify both the controller and the portID, which represent the two ends of the fabric:

```
# vxdmpadm enable ctlr=ctlr_name enclosure=enclr_name \
  portid=array_port_ID
```

To enable I/O for a particular DMP node, specify the DMP node name.

```
# vxdmpadm enable dmpnodename=dmpnode
```

Renaming an enclosure

The `vxdmpadm setattr` command can be used to assign a meaningful name to an existing enclosure, for example:

```
# vxdmpadm setattr enclosure emc0 name=GRP1
```

This example changes the name of an enclosure from `emc0` 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    ARRAY_TYPE LUN_COUNT F
==========================================================================
Disk            Disk           DISKS         CONNECTED Disk       6         -
GRP1            EMC            000292601383  CONNECTED A/A        1         5
hitachi_usp-vm0 Hitachi_USP-VM 25847         CONNECTED A/A        1         6
emc_clariion0   EMC_CLARiiON   CK20007040035 CONNECTED CLR-A/PF   2         0
```

Configuring the response to I/O failures

You can configure how Dynamic Multi-Pathing (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 minutes on various active paths.

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 81.

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 for recovery.

For example, the following command sets the default settings:

```
# vxdmpadm setattr arraytype A/A recoveryoption=default
```

For PCI devices, the default settings are `recoveryoption=fixedretry retrycount=5`.

For all other devices, the default settings are `recoveryoption=timebound iotimeout=300`

Specifying `recoveryoption=default`also has the effect of configuring I/O throttling with the default settings.

See "Configuring the I/O throttling mechanism" on page 79.

Note: The response to I/O failure settings is persistent across reboots of the system.

Configuring the I/O throttling mechanism

By default, Dynamic Multi-Pathing (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 81.

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 77.

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`. The default value of the tunable is `1`, which represents that the feature is on.

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 that triggers SFG.

```
# vxdmpadm settune dmp_sfg_threshold=N
```

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 (LIPP)

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 the 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 Dynamic Multi-Pathing (DMP) 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]   Nothrottle[0]
HDS9500-ALUA0   Error-Retry      Timebound[300]  Timebound[300]
```

The command output shows the default and current policy options and their values.

Table 3-3 summarizes the possible recovery option settings for retrying I/O after an error.

Table 3-3 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 3-4 summarizes the possible recovery option settings for throttling I/O.

Table 3-4 Recovery options for I/O throttling

Recovery option	Possible settings	Description
recoveryoption=nothrottle	None	I/O throttling is not used.

Table 3-4 Recovery options for I/O throttling *(continued)*

Recovery option	Possible settings	Description
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

Dynamic Multi-Pathing (DMP) maintains a kernel task 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 task 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 83.

Use the `vxdmpadm settune dmp_restore_policy` command to configure one of the following restore policies. The policy remains in effect until the restore thread is stopped or the values are changed using the `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 settune dmp_restore_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 settune dmp_restore_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 settune dmp_restore_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 a large number of paths are available. The command to configure this policy is:

  ```
  # vxdmpadm settune dmp_restore_policy=check_periodic
  ```

  The default number of cycles between running the `check_all` policy is 10.

The `dmp_restore_interval` tunable parameter specifies how often the path restoration thread examines the paths. For example, the following command sets the polling interval to 400 seconds:

```
# vxdmpadm settune dmp_restore_interval=400
```

The settings are immediately applied and are persistent across reboots. Use the `vxdmpadm gettune` command to view the current settings.

See "DMP tunable parameters" on page 147.

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 Dynamic Multi-Pathing (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 `vxdmpadm gettune` command to display the tunable parameter values that show the status of the Dynamic Multi-Pathing (DMP) path restoration thread. These tunables include:

dmp_restore_state	the status of the automatic path restoration kernel thread.
dmp_restore_interval	the polling interval for the DMP path restoration thread.
dmp_restore_policy	the policy that DMP uses to check the condition of paths.

To display the status of the DMP path restoration thread

- Use the following commands:

```
# vxdmpadm gettune dmp_restore_state

# vxdmpadm gettune dmp_restore_interval

# vxdmpadm gettune dmp_restore_policy
```

Configuring Array Policy Modules

Dynamic Multi-Pathing (DMP) provides Array Policy Modules (APMs) for use with an array. An APM is a dynamically loadable kernel module (or plug-in) that 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 DMP provides, or that another version of the APM provides.

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 from 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.

Specify the `-r` option to remove an APM that is not currently loaded:

```
# vxdmpadm -r cfgapm module_name
```

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

Configuring latency threshold tunable for metro/geo array

The Metro/Geo cluster system has visibility for both paths, local and remote. Dynamic Multi-Pathing (DMP) A/A array IO policy distributes IO evenly across both of these paths, local and remote. Also, Metro/Geo support serves IO through the local path if the remote paths have higher latency than latency_threshold_difference.

Tunable lets you manage the parameter values of latency_threshold_difference. The default value of the latency_threshold_difference of the tunable parameter is 200 µs. The latency_threshold_difference tunable is valid only for the METRO type of array.

You can assign a new latency threshold difference value to a specified enclosure. Using tunable option you can increase or decrease the latency_threshold_difference value.

The following is the syntax for the tuning latency_threshold_difference:

```
# vxdmpadm getattr {enclosure enc-name}
\  latency_threshold_difference
```

Use the following command `vxdmpadm getattr` that displays the latency_threshold_difference value that is set for an enclosure **emc0**:

```
# vxdmpadm getattr enclosure emc0 latency_threshold_difference
ENCLR_NAME  default  CURRENT
============================================
emc0        200      200
```

To set new value, you can use the following syntax for the specified enclosure:

```
#  vxdmpadm setattr enclosure <enclr-name>
{name=<value> | \  latency_threshold_difference=<value>}
```

The following example shows the `vxdmpadm setattr` command to set the latency_threshold_difference value that is set for an enclosure **emc0:**

```
#  vxdmpadm setattr enclosure emc0 latency_threshold_difference=300
```

Note: The latency_threshold_difference is applicable for enclosure only.

DMP serves IO with local paths only if remote path has higher IO latency, then marks it as STANDBY.

```
#vxdmpadm getsubpaths dmpnodename=emc0_00d2
```

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

```
NAME STATE[A]   PATH-TYPE[M] CLTR-NAME ENCLR-TYPE ENCLR-NAME ATTRS  PRIORITY
=============================================================================
Sdar ENABLED       -           c9      EMC         emc0     STANDBY    -
Sdbj ENABLED(A)    -           c9      EMC         emc0        -       -
Sdh  ENABLED(A)    -           c9      EMC         emc0        -       -
Sdz  ENABLED       -           c9      EMC         emc0     STANDBY    -
```

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

Chapter 4

Administering disks

This chapter includes the following topics:

About disk management

Dynamic Multi-Pathing (DMP) is used to administer multiported disk arrays.

See "How DMP works" on page 9.

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 91.

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 Dynamic Multi-Pathing (DMP) internal database to reflect the new state of the system.

To reconfigure the DMP database, first make Linux 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 paths to a physical disk 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 `sdm` and `sdn`:

```
# vxdisk scandisks device=sdm,sdn
```

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=sdm,sdn
```

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=c1,c2
```

The next command discovers only those devices that are connected to the specified physical controller:

```
# vxdisk scandisks pctlr=c1+c2
```

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.

About discovering disks and dynamically adding disk arrays

Dynamic Multi-Pathing (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, Dynamic Multi-Pathing (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 plug-ins that allow close tie-in of DMP with any specific array model.

Refer to the Hardware Compatibility List at:

https://www.veritas.com/content/support/en_US/doc/infoscale_hcl_8x_win

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 100.

DMP can provide basic multi-pathing to arrays that comply with the Asymmetric Logical Unit Access (ALUA) standard, 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 Dynamic Multi-Pathing (DMP) 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 that are capable of being multi-pathed 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 100.

Disks in JBODs that do not fall into any supported category, and which are not capable of being multi-pathed by DMP are placed in the `OTHER_DISKS` category.

Adding DMP 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 Veritas. Veritas provides support for new disk arrays through updates to the `VRTSaslapm` rpm. To determine if an updated `VRTSaslapm` rpm is available for download, refer to the hardware compatibility list (HCL). The hardware compatibility list provides a link to the latest rpm for download and instructions for installing the `VRTSaslapm` rpm. You can upgrade the `VRTSaslapm` rpm while the system is online; you do not need to stop the applications.

Refer to the hardware compatibility list at:

https://www.veritas.com/content/support/en_US/doc/infoscale_hcl_8x_unix

Each VRTSaslapm rpm is specific for the Dynamic Multi-Pathing version. Be sure to install the `VRTSaslapm` rpm that supports the installed version of Dynamic Multi-Pathing.

The new disk array does not need to be already connected to the system when the `VRTSaslapm` rpm is installed.

If you need to remove the latest `VRTSaslapm` rpm, you can revert to the previously installed version. For the detailed procedure, refer to the *Veritas InfoScale 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
```

About third-party driver coexistence

The third-party driver (TPD) coexistence feature of Dynamic Multi-Pathing (DMP) allows I/O that is controlled by some third-party multi-pathing drivers to bypass Dynamic Multi-Pathing (DMP) while retaining the monitoring capabilities of DMP. If a suitable Array Support Library (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. The TPD coexistence feature of DMP permits coexistence without requiring any change in a third-party multi-pathing driver.

See "Changing device naming for enclosures controlled by third-party drivers" on page 107.

See "Displaying information about devices controlled by third-party drivers" on page 55.

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 Dynamic Multi-Pathing (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 claimed 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)
           Device sda
. . .
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)
           Device sdb
           Device sdc
       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 (HBAs) configured on the system, including iSCSI adapters.

Table 4-1 shows the HBA information.

Table 4-1 HBA information

Field	Description
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

PORT-ID   HBA-ID    STATE      ADDRESS
-----------------------------------------------------------
c2_p0     c2        Online     50:0A:09:80:85:84:9D:84
c3_p0     c3        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.

Table 4-2 shows the target information.

Table 4-2 Target information

Field	Description
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:

```
TARGET-ID  ALIAS HBA-ID STATE  ADDRESS
-----------------------------------------------------------------
c2_p0_t0   -     c2     Online 50:0A:09:80:85:84:9D:84
c3_p0_t1   -     c3     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=c2
```

```
TARGET-ID  ALIAS  HBA-ID   STATE      ADDRES
-----------------------------------------------------------------
c2_p0_t0   -      c2       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.

Table 4-3 shows the device information.

Table 4-3 Device information

Field	Description
Device	The device name.
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)
-------------------------------------------------------
sda       fscsi0_p0_t0 Online   CLAIMED (libvxemc.so)
sdb       fscsi0_p0_t0 Online   SKIPPED (libvxemc.so)
sdc       fscsi0_p0_t0 Offline  ERROR
sdd       fscsi0_p0_t0 Online   EXCLUDED
sde       fscsi0_p0_t0 Offline  MASKED
nvme0n1   -            Online   CLAIMED (libvxnvme_nonscsi.so)
```

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 4-4 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

Table 4-4 Parameters for iSCSI devices *(continued)*

Parameter	Default value	Minimum value	Maximum value
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.

  ```
  # vxddladm getiscsi target=c2_p2_t0
  ```

  The following is a sample output:

  ```
  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

- Use the following command:

```
# vxddladm listsupport all
```

Excluding support for a disk array library

You can exclude support for disk arrays that depends on a particular disk array library. You can also exclude support for disk arrays from a particular vendor.

To exclude support for a disk array library

- To exclude support for a disk array library, specify the array library to the following command.

```
# vxddladm excludearray libname=libname
```

You can also exclude support for disk arrays from a particular vendor, as shown in this example:

```
# vxddladm excludearray vid=ACME pid=X1

# vxdisk scandisks
```

Re-including support for an excluded disk array library

If you previously excluded support for all arrays that depend on a particular disk array library, use this procedure to include the support for those arrays. This procedure removes the library from the exclude list.

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.

```
# vxddladm includearray libname=libname
```

This command adds the array library to the database so that the library can once again be used in device discovery.

```
# vxdisk scandisks
```

Listing excluded disk arrays

To list all disk arrays that are currently excluded from use by Veritas Volume Manager (VxVM)

- Type the following command:

```
# vxddladm listexclude
```

Listing disks claimed in the DISKS category

To list disks that are claimed in the `DISKS` (JBOD) category

- Type the following command:

```
# vxddladm listjbod
```

Displaying details about an Array Support Library

Dynamic Multi-Pathing (DMP) enables you to display details about the Array Support Libraries (ASL).

To display details about an Array Support Library

- Type the following command:

```
# vxddladm listsupport libname=library_name.so
```

This command displays the vendor IDs (`VIDs`), product IDs (`PIDs`) for the arrays, array types (for example, A/A or A/P), and array names. The following is sample output.

```
# vxddladm listsupport libname=libvxfujitsu.so
ATTR_NAME            ATTR_VALUE
==============================================
LIBNAME              libvxfujitsu.so
VID                  vendor
PID                  GR710, GR720, GR730
                     GR740, GR820, GR840
ARRAY_TYPE           A/A, A/P
ARRAY_NAME           FJ_GR710, FJ_GR720, FJ_GR730
                     FJ_GR740, FJ_GR820, FJ_GR840
```

Adding unsupported disk arrays to the DISKS category

Disk arrays should be added as JBOD devices if no Array Support Library (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 9.

Warning: This procedure ensures that Dynamic Multi-Pathing (DMP) is set up correctly on an array that is not supported by Veritas Volume Manager (VxVM). Otherwise, VxVM 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 output shows that the vendor ID is `SEAGATE` and the product ID is `ST318404LSUN18G`.

   ```
   Vendor id (VID)      : SEAGATE
   Product id (PID)     : ST318404LSUN18G
   Revision             : 8507
   Serial Number        : 0025T0LA3H
   ```

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 `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 91.

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 FIRMWARE
============================================================================
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
punr710vm04_disk_1 auto:none       -            -            online invalid
punr710vm04_disk_2 auto:none       -            -            online invalid
punr710vm04_disk_3 auto:none       -            -            online invalid
punr710vm04_disk_4 auto:none       -            -            online invalid
sda                auto:none       -            -            online invalid
xiv0_9148          auto:none       -            -            online invalid thinrclm
...
```

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
============================================================================
punr710vm04_disk_1   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_2   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_3   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_4   ENABLED      Disk         1      1     0     disk
sda                  ENABLED      Disk         1      1     0     disk
...
```

The output in this example 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

Use the procedure in this section to remove 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 that have the vendor id of SEAGATE:

```
# vxddladm rmjbod vid=SEAGATE
```

Foreign devices

The Device Discovery Layer (DDL) may not be able to discover some devices that are not auto-discoverable, such as RAM disks. Such foreign devices can be made available as simple disks to Veritas Volume Manager (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
```

If a block or character device is not supported by a driver, it can be omitted from the command as shown here:

```
# vxddladm addforeign blockdir=/dev/foo/dsk
```

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.

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.

Changing the disk device 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 Dynamic Multi-Pathing (DMP) with native volumes, the disk naming scheme must be EBN, the `use_avid` attribute must be `yes`, 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.

  OR

  Change the naming scheme from the command line. Use the following command to select enclosure-based naming:

  ```
  # vxddladm set namingscheme=ebn [persistence={yes|no}] \
  [use_avid={yes|no}] [lowercase={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 lowercase. 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 106.

Displaying the disk-naming scheme

In Dynamic Multi-Pathing (DMP), disk naming can be operating system-based naming or enclosure-based naming.

The following 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
NAMING_SCHEME    PERSISTENCE LOWERCASE USE_AVID
=============================================
Enclosure Based  Yes         Yes       Yes
```

See "Disk device naming in DMP" on page 18.

Regenerating persistent device names

The persistent device naming feature makes the names of disk devices persistent across system reboots. The Device Discovery Layer (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 operating system-based and enclosure-based names are regenerated.

Changing device naming for enclosures controlled by third-party drivers

By default, enclosures controlled by third-party drivers (TPD) 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 Dynamic Multi-Pathing (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 105.

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 Array Support Library (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 `pp_emc_clariion0`. In this example, the device-naming scheme is set to OSN.

  ```
  # vxdisk list

  DEVICE       TYPE            DISK         GROUP        STATUS
  emcpowerp    auto:cdsdisk    -            -            online
  emcpowerq    auto:cdsdisk    -            -            online
  emcpowerr    auto:cdsdisk    -            -            online
  emcpowers    auto:cdsdisk    -            -            online
  emcpowert    auto:cdsdisk    -            -            online

  # vxdmpadm setattr enclosure pp_emc_clariion0 tpdmode=native

  # vxdisk list

  DEVICE       TYPE            DISK         GROUP        STATUS
  sde          auto:cdsdisk    -            -            online
  sdf          auto:cdsdisk    -            -            online
  sdg          auto:cdsdisk    -            -            online
  sdh          auto:cdsdisk    -            -            online
  sdi          auto:cdsdisk    -            -            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 5

Dynamic Reconfiguration of devices

This chapter includes the following topics:

About online Dynamic Reconfiguration

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.

Note: You can change the LUN configuration dynamically either using the Dynamic Reconfiguration (DR) tool or manually. Veritas recommends using the Dynamic Reconfiguration tool.

Table 5-1 lists the kinds of online dynamic reconfigurations that you can perform:

Table 5-1

Task	Topic
Reconfigure a LUN online that is under DMP control	■ DR tool—See "Reconfiguring a LUN online that is under DMP control using the Dynamic Reconfiguration tool" on page 112. ■ Manual—See "Manually reconfiguring a LUN online that is under DMP control" on page 121.
Replace a host bus adapter (HBA) online	■ DR tool—See "Replacing a host bus adapter online" on page 120.
Update the array controller firmware, also known as a nondisruptive upgrade	■ See "Upgrading the array controller firmware online" on page 131.

About the DMPDR utility

Adding and removing storage logical unit number (LUNs) on a server is a complex process that involves a series of steps that must be performed in the correct order. Any incorrect step can lead to errors and may require a system restart to recover from that error.

InfoScale provides the Dynamic Multipathing Dynamic Reconfiguration (DMPDR) utility that automates the LUN reconfiguration process. DMPDR provides a non-interactive interface that lets you perform the steps in the correct sequence thereby reducing human error and a need for manual intervention.

DMPDR performs the following tasks:

- Runs the correct sequence of steps to detect any newly added and removed LUNs.
- Updates the underlying OS device handles for the DMP-managed devices and refreshes the VxVM and the DMP structures.
- Creates a log file under `/var/adm/vx/` for each LUN reconfiguration event.

Considerations for using DMPDR

- The DMPDR utility works only with InfoScale Dynamic Multipathing. Any third-party device drivers (such as MPxIO and EMC PowerPath) are not supported and do not work with the DMPDR interface.
- You must run separate instances or sessions of the DMPDR utility when you physically add or remove LUNs. Do not perform add and remove LUN operations in the same session as it may cause issues with the I/O stack.

Accessing the DMPDR utility

The DMPDR utility is located at `/usr/lib/vxvm/voladm.d/bin/dmpdr`. After you install InfoScale, a soft link to the utility is created at `/opt/VRTS/bin` for easy access.

Run the DMPDR utility before and after any add or remove LUN operations to ensure that the correct sequence of commands is performed.

Type the following command to run the DMPDR utility:

```
# /usr/lib/vxvm/voladm.d/bin/dmpdr – o refresh
```

Using the DMPDR utility to reconfigure the LUNs associated with a server

The following procedure describes how to use DMPDR to reconfigure LUNs on a server after they are added or removed from the underlying storage layer.

Reconfiguring LUN removal

Perform the following steps:

1. Inform Volume Manager (VxVM) about a change in the disk access name before you remove any LUNs.

   Type the following command:

   ```
   # vxdisk rm diskdevicename
   ```

2. Remove the LUNs from the storage array.
3. Run the DMPDR utility.

   Type the following command:

   ```
   # /usr/lib/vxvm/voladm.d/bin/dmpdr -o refresh
   ```

   The command output displays the progress of the entire process. After the process completes successfully, you can proceed to add or reconfigure new LUNs.

Note: The SCSI Peripheral Qualifier/Device Type (PQ) attribute reports a non-zero value for LUNs that have been removed intentionally from the storage layer.

For example, here's an inquiry for a device named emc0_00aa:

```
# /etc/vx/diag.d/vxscsiinq -d /dev/vx/dmp/emc0_00aa | head -3
```

The inquiry command output shows the following:

```
Inquiry for /dev/vx/dmp/emc0_00aa, evpd 0x0, page code 0x0 Peripheral
Qualifier/Device Type : 3f
```

The DMPDR tool automatically deletes only the stale OS device handles that show a non-zero value for the Peripheral Qualifier/Device Type (PQ) attribute.

Reconfiguring LUN addition

Perform the following steps:

1. Add the LUNs from the storage array.
2. Run the DMPDR utility.

   Type the following command:

   ```
   # /usr/lib/vxvm/voladm.d/bin/dmpdr -o refresh
   ```

   The command output displays the progress of the entire process. The DMPDR utility creates a log file at `/var/adm/vx/` for each LUN reconfiguration operation.

Reconfiguring a LUN online that is under DMP control using the Dynamic Reconfiguration tool

Perform the following tasks to reconfigure a LUN online that is under DMP control using the Dynamic Reconfiguration tool:

Table 5-2

Task	Topic
Removing LUNs dynamically from an existing target ID	See "Removing LUNs dynamically from an existing target ID" on page 113.
Adding LUNs dynamically to a new target ID	See "Adding new LUNs dynamically to a target ID" on page 116.
Replacing a LUN on an existing target ID	See "Replacing LUNs dynamically from an existing target ID" on page 119.

Table 5-2 *(continued)*

Task	Topic
Changing the LUN characteristics	See "Changing the characteristics of a LUN from the array side" on page 130.

Removing LUNs dynamically from an existing target ID

Dynamic Multipathing (DMP) provides a Dynamic Reconfiguration tool to simplify the removal of LUNs from an existing target ID. Each LUN is unmapped from the host. DMP issues an operating system device scan and cleans up the operating system device tree.

Warning: Do not run any device discovery operations outside of the Dynamic Reconfiguration tool until the device operation is completed.

In a cluster, perform the steps on all nodes in the cluster.

To remove LUNs dynamically from an existing target ID

1. Stop all applications and volumes that are hosted on the LUNs that are to be removed.

   For LUNs using Linux LVM over DMP devices, remove the device from the LVM volume group.

   ```
   # vgreduce vgname devicepath
   ```

2. Start the `vxdiskadm` utility:

   ```
   # vxdiskadm
   ```

3. Select the **Dynamic Reconfiguration operations** option from the `vxdiskadm` menu.
4. Select the **Remove LUNs** option.

5 Type **list** or press **Return** to display a list of LUNs that are available for removal. A LUN is available for removal if it is not in use.

The following shows an example output:

```
Select disk devices to remove: [<pattern-list>,all,list]: list
LUN(s) available for removal:
eva4k6k0_0
eva4k6k0_1
eva4k6k0_2
eva4k6k0_3
eva4k6k0_4
emc0_0119
```

6 Enter the name of a LUN, a comma-separated list of LUNs, or a regular expression to specify the LUNs to remove.

For example, enter `emc0_0119`.

```
Select disk devices to Remove: [<pattern-list>,all,list,
file=<filename>,q] (default:list): emc0_0119
```

7 At the prompt, confirm the LUN selection.

DMP removes the LUN from VxVM usage.

8 At the following prompt, remove the LUN from the array or the target.

```
Remove Luns
Menu: VolumeManager/Disk/DynamicReconfigurationOperations/RemoveLuns

INFO: Removing Lun [emc0_0119] from VxVM
INFO: LUN [emc0_0119] removed successfully from VxVM.
-----------------------------------------------------
Enclosure=emc0 AVID=0119
Device=emc0_0119 Serial=2200119000
PATH=sdad ctlr=c11 port=16c-0 [-]
PATH=sdah ctlr=c12 port=16c-0 [-]
PATH=sdaj ctlr=c12 port=16c-1 [-]
PATH=sdaf ctlr=c11 port=16c-1 [-]
-----------------------------------------------------
Please remove LUNs with Above details from array and press 'y' to
continue removal or 'q' to quit :
```

9 The following are sample EMC Symmetrix commands:

```
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 0 remove devs 0119
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 1 remove devs 0119
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 0 remove devs 0119
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 1 remove dev 0119

# symmask -sid 822 refresh -nopr

Symmetrix FA/SE directors updated with contents of SymMask
Database 000290300822

When complete, respond to previous array prompt.

Please remove LUNs with Above details from array and
press 'y' to continue removal or 'q' to quit : y
```

10 DMP completes the removal of the device from VxVM usage. Output similar to the following is displayed:

```
Remove Luns
Menu: VolumeManager/Disk/DynamicReconfigurationOperations/RemoveLuns

INFO: Checking/Removing stale device entries (if any).
INFO: Refreshing OS device Tree
INFO: Updating VxVM device tree
------------------------------------------------------------
Luns Removed
------------------------------------------------------------
emc0_0119
------------------------------------------------------------

Press <Enter> or <Return> to continue:
```

11 Specify the dynamic reconfiguration operation to be done:

```
Specify Dynamic Reconfiguration Operation to be done:
Menu: VolumeManager/Disk/DynamicReconfigurationOperations

 1 Add Luns
 2 Remove Luns
 3 Replace Luns
 4 Replace HBA


 ? Display help about menu
 ?? Display help about the menuing system
 q Exit
```

To exit the Dynamic Reconfiguration tool, enter: `q`

Adding new LUNs dynamically to a target ID

Dynamic Multi-Pathing (DMP) provides a Dynamic Reconfiguration tool to simplify the addition of new LUNs to a new or existing target ID. One or more new LUNs are mapped to the host by way of multiple HBA ports. An operating system device scan is issued for the LUNs to be recognized and added to DMP control.

Warning: Do not run any device discovery operations outside of the Dynamic Reconfiguration tool until the device operation is completed.

In a cluster, perform the steps on all the nodes in the cluster.

To add new LUNs dynamically to a target ID

1. Start the `vxdiskadm` utility:

   ```
   # vxdiskadm
   ```

2. Select the **Dynamic Reconfiguration operations** option from the `vxdiskadm` menu.

3. Select the **Add LUNs** option.

   Output similar to the following is displayed:

   ```
   Add Luns
   Menu: VolumeManager/Disk/DynamicReconfigurationOperations/AddLuns

   INFO: Refreshing OS device Tree
   INFO: Updating VxVM device tree
   Add LUNs from array, once done then press 'y' to continue
   or 'q' to quit. :
   ```

4. The following are sample EMC Symmetrix commands:

   ```
   # symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 0 add devs
   0119 -nopr
   # symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 1 add devs
   0119 -nopr
   # symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 0 add devs
   0119 -nopr
   # symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 1 add devs
   0119 -nopr
   # symmask -sid 822 refresh -nopr

   Symmetrix FA/SE directors updated with contents of SymMask
   Database 000290300822
   ```

5 When the prompt displays, add the LUNs from the array.

Output similar to the following is displayed:

```
Add LUNs from array, once done then press 'y' to continue
or 'q' to quit. : y

Add Luns
Menu: VolumeManager/Disk/DynamicReconfigurationOperations/AddLuns

INFO: Refreshing OS device Tree
INFO: Updating VxVM device tree
INFO: Updating partition table information and disk size
INFO: Number of Paths for Lun [emc0_0119] presented=4
INFO: Updating VxVM device tree
```

6 Select **y** to continue to add the LUNs to DMP.

DMP updates the operating system device tree and the VxVM device tree. The newly-discovered devices are now visible.

```
-----------------------------------------------------------
Luns Added
-----------------------------------------------------------
Enclosure=emc0 AVID=0119
Device=emc0_0119 Serial=2200119000
PATH=sdaf ctlr=c11 port=16c-1 [-]
PATH=sdah ctlr=c12 port=16c-0 [-]
PATH=sdaj ctlr=c12 port=16c-1 [-]
PATH=sdad ctlr=c11 port=16c-0 [-]
-----------------------------------------------------------

Press <Enter> or <Return> to continue:
```

7 Specify the dynamic reconfiguration operation to be done:

```
Specify Dynamic Reconfiguration Operation to be done:
Menu: VolumeManager/Disk/DynamicReconfigurationOperations

 1 Add Luns
 2 Remove Luns
 3 Replace Luns
 4 Replace HBA


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

Select an operation to perform : q
```

To exit the Dynamic Reconfiguration tool, enter: `q`

Replacing LUNs dynamically from an existing target ID

Dynamic Multipathing (DMP) provides a Dynamic Reconfiguration tool to simplify the replacement of new LUNs from an existing target ID. Each LUN is unmapped from the host. DMP issues an operating system device scan and cleans up the operating system device tree.

Warning: Do not run any device discovery operations outside of the Dynamic Reconfiguration tool until the device operation is completed.

In a cluster, perform the steps on all the nodes in the cluster.

To replace LUNs dynamically from an existing target ID

1. Stop all applications and volumes that are hosted on the LUNs that are to be removed.

   For LUNs using Linux LVM over DMP devices, remove the device from the LVM volume group

   ```
   # vgreduce vgname devicepath
   ```

2. Start the `vxdiskadm` utility:

   ```
   # vxdiskadm
   ```

3. Select the **Dynamic Reconfiguration operations** option from the `vxdiskadm` menu.
4. Select the **Replace LUNs** option.

   The output displays a list of LUNs that are available for replacement. A LUN is available for replacement if there is no open on the LUN, and the state is online or nolabel.
5. Select one or more LUNs to replace.
6. At the prompt, confirm the LUN selection.
7. Remove the LUN from the array or the target.
8. Return to the Dynamic Reconfiguration tool and select `y` to continue the removal.

   After the removal completes successfully, the Dynamic Reconfiguration tool prompts you to add a LUN.
9. When the prompt displays, add the LUNs from the array or the target.
10. Select **y** to continue to add the LUNs.

    DMP updates the operating system device tree and the VxVM device tree. The newly-discovered devices are now visible.

Replacing a host bus adapter online

Dynamic Multi-Pathing (DMP) provides a Dynamic Reconfiguration tool to simplify the removal of host bus adapters from an existing system.

To replace a host bus adapter online

1 Start the `vxdiskadm` utility:

```
# vxdiskadm
```

2 Select the **Dynamic Reconfiguration operations** option from the `vxdiskadm` menu.

3 Select the **Replace HBAs** option.

The output displays a list of HBAs that are available to DMP.

4 Select one or more HBAs to replace.

5 At the prompt, confirm the HBA selection.

6 Replace the host bus adapter.

7 Return to the Dynamic Reconfiguration tool and select `y` to continue the replacement process.

DMP updates the operating system device tree.

Manually reconfiguring a LUN online that is under DMP control

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.

Overview of manually reconfiguring a LUN

This section only provides an overview of the prechecks and the procedure to manually add or remove a LUN. The procedures have been elaborately documented in the topics listed in the following table:

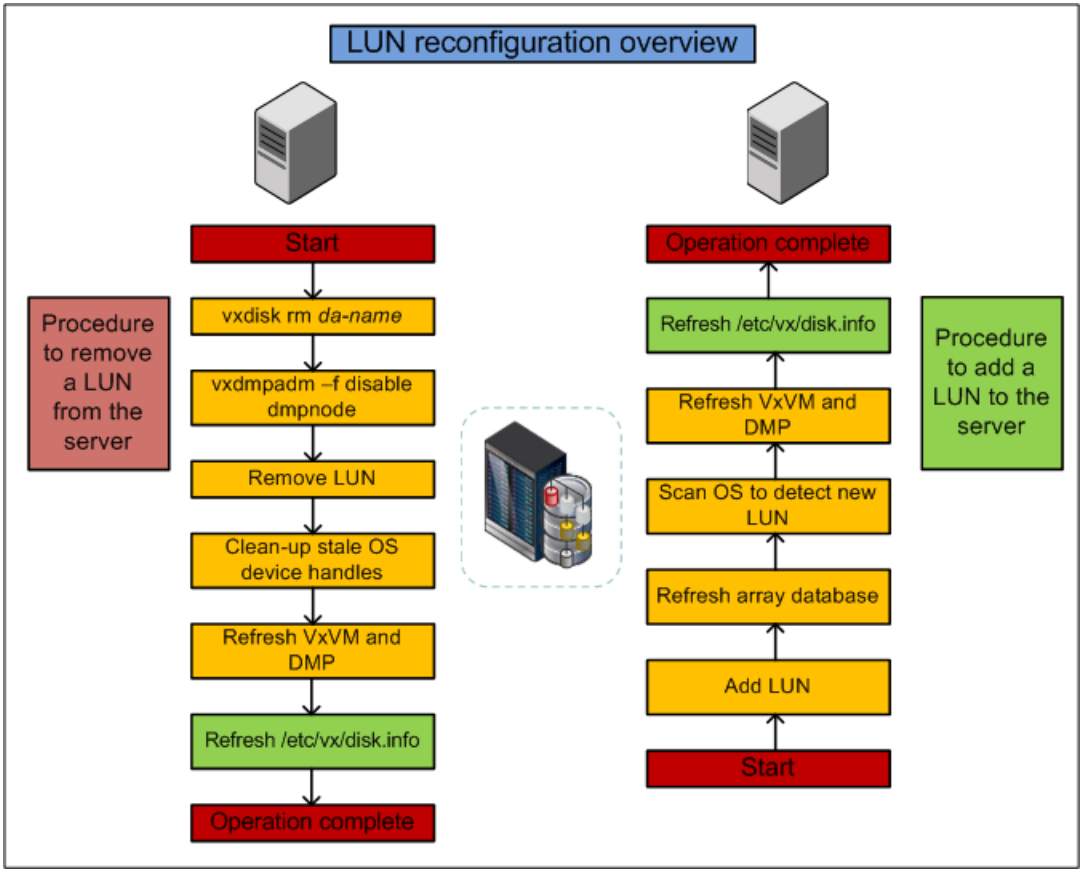
Table 5-3

Task	Topic
Removing LUN dynamically from an existing target ID	See "Manually removing LUNs dynamically from an existing target ID" on page 124.
Cleaning up the operating system device tree after removing LUNs	See "Manually cleaning up the operating system device tree after removing LUNs" on page 129.

Table 5-3 *(continued)*

Task	Topic
Scanning an operating system device tree after adding or removing LUNs	See "Scanning an operating system device tree after adding or removing LUNs" on page 128.
Adding LUN dynamically to a new target ID	See "Manually adding new LUNs dynamically to a new target ID" on page 127.
Changing the LUN characteristics	See "Changing the characteristics of a LUN from the array side" on page 130.

Figure 5-1 LUN reconfiguration overview



Prechecks

Perform the following prechecks before manually reconfiguring a LUN:

Table 5-4 Prechecks

Task	Command
Check the `/etc/vx/disk.info` file	`# grep "0xffff" /etc/vx/disk.info`
Refresh the OS layer	`# echo '- - -' > /sys/class/scsi_host/host$i/scan`
List OS device handles	`# lsscsi`
Refresh VxVM and DMP	`# vxdisk scandisks`
Refresh DDL layer/dev_t (device number) list	`# vxddladm assign names`

Note: Ensure that the OS and VxVM are both clean prior to provisioning any new LUNs.

Manually removing a LUN

Perform the following steps to manually remove a LUN:

Table 5-5 LUN removal steps

Task	Validation
Unmount file system (s)	Confirm whether the disk has been removed from the disk group.
Close the VxVM device: `# vxdisk rm da-name`	Confirm whether the VxVM device has been closed: `# vxdisk list`
Disable DMP paths: `# vxdmpadm –f disable dmpnodename=da-name`	Confirm whether the DMP paths have been disabled: `# vxdmpadm getsubpaths dmpnodename=da-name`
Mask LUN from the server	Confirm whether the LUN has been removed at the array level.

Table 5-5 LUN removal steps *(continued)*

Task	Validation
Clean-up OS device handles: `# echo 1 > /sys/block/device_name/device/delete`	Confirm whether OS device handles are clean: `# lsscsi`
Refresh VxVM and DMP: `# vxdisk scandisks`	
Refresh DDL layer/dev_t (device number) list: `# vxddladm assign names`	

Manually adding a LUN

To manually add a LUN

1. Mask LUN to HBA worldwide name (WWN) in the server.
2. Refresh the array database.
3. Refresh OS device handles.
4. Refresh VxVM and DMP.
5. Refresh the `/etc/vx/disk.info` file.

Manually 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 clean up the operating system device tree.

The high-level procedure and the DMP commands are generic. However, the operating system commands may vary depending on the Linux version. For example, the following procedure uses Linux Suse10.

To remove LUNs dynamically from an existing target ID

1 Before any dynamic reconfiguration, ensure that the `dmp_cache_open` tunable is set to `on`. This setting is the default.

```
# vxdmpadm gettune dmp_cache_open
```

If the tunable is set to `off`, set the `dmp_cache_open` tunable to `on`.

```
# vxdmpadm settune dmp_cache_open=on
```

2 Identify which LUNs to remove from the host. Do one of the following:

- Use the 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.

3 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 a disk group, remove the LUNs from the disk group using the following command. 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
  ```

4 For LUNs using Linux LVM over DMP devices, remove the device from the LVM volume group.

```
# vgreduce vgname devicepath
```

5 Using the AVID or the LUN index, use the Storage Array Management to unmap or unmask the LUNs you identified in step 2.

6 Remove the LUNs from the `vxdisk` list. Enter the following command on all nodes in a cluster:

```
# vxdisk rm da-name
```

This step is required. If you do not perform this step, the DMP device tree shows ghost paths.

7 Clean up the Linux SCSI device tree for the devices that you removed in step 6.

See “Manually cleaning up the operating system device tree after removing LUNs” on page 129.

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.

8 Scan the operating system device tree.

See “Scanning an operating system device tree after adding or removing LUNs” on page 128.

9 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`

10 Refresh the DMP device name database using the following command:

```
# vxddladm assign names
```

11 Verify that the LUNs were removed cleanly by answering the following questions:

- Is the device tree clean?
  Verify that the operating system metanodes are removed from the `/sys/block` directory.
- 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 5 and perform the required steps.

If the answer to all of the questions is "Yes," the LUN remove operation is successful.

Manually 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. However, the operating system commands may vary depending on the Linux version. For example, the following procedure uses Linux Suse10.

To add new LUNs dynamically to a new target ID

1. Prior to any dynamic reconfiguration, ensure that the `dmp_cache_open` tunable is set to `on`. This setting is the default.

   ```
   # vxdmpadm gettune dmp_cache_open
   ```

   If the tunable is set to `off`, set the `dmp_cache_open` tunable to `on`.

   ```
   # vxdmpadm settune dmp_cache_open=on
   ```

2. 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.
3. Map/mask the LUNs to the new target IDs on multiple hosts.
4. Scan the operating system device.

   See "Scanning an operating system device tree after adding or removing LUNs" on page 128.

   Repeat step 2 and step 3 until you see that all the LUNs have been added.
5. 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`
6. Refresh the DMP device name database using the following command:

   ```
   # vxddladm assign names
   ```

7. 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 2 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

If you try to reprovision a LUN or set of LUNs whose previously-valid operating system device entries are not cleaned up, the following messages are displayed. Also, DMP reconfiguration during the DMP device scan and DMP reconfiguration are temporarily inhibited.

See "Manually cleaning up the operating system device tree after removing LUNs" on page 129.

```
VxVM vxdisk ERROR V-5-1-14519 Data Corruption Protection Activated
- User Corrective Action Needed

VxVM vxdisk INFO V-5-1-14521 To recover, first ensure that the OS
device tree is up to date (requires OS specific commands).

VxVM vxdisk INFO V-5-1-14520 Then, execute 'vxdisk rm' on the
following devices before reinitiating device discovery. <DA names>
```

The message above indicates that a new LUN is trying to reuse the target ID of an older LUN. The device entries have not been cleaned, so the new LUN cannot use the target ID. Until the operating system device tree is cleaned up, DMP prevents this operation.

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.

Linux provides several methods for rescanning the SCSI bus and identifying the devices mapped to it. These methods include the following:

- The SCSI scan function in the `/sys` directory

- HBA vendor utilities

To scan using the SCSI scan function

- Enter the following command:

  ```
  # echo '- - -' > /sys/class/scsi_host/host$i/scan
  ```

  where the three dashes refer to the channel, target, and LUN numbers, and *host$i* is the host bus adapter instance. This example scans every channel, target, and LUN visible via this host bus adapter instance.

To scan using HBA vendor utilities

- Follow the vendor's instructions for the HBA utility. Examples include the following:
  - QLogic provides a script that dynamically scans for newly-added LUNs. You can download it from the QLogic Web site. To run the script, enter the following command:

    ```
    # ./ql-dynamic-tgt-lun-disc.sh
    ```

  - Emulex provides an HBAnywhere script. You can download it from the Emulex web site. The script has a LUN Scan Utility that dynamically scans for newly-added LUNs. To run the utility, enter the following command:

    ```
    # lun_scan all
    ```

Manually cleaning up the operating system device tree after removing LUNs

After you remove LUNs, you must clean up the operating system device tree.

The operating system commands may vary, depending on the Linux version. The following procedure uses SUSE 10. If any of these steps do not produce the desired result, contact Novell support.

To clean up the operating system device tree after removing LUNs

1. Remove the device from the operating system database. Enter the following command:

   ```
   # echo 1 > /sys/block/$PATH_SYS/device/delete
   ```

   where *PATH_SYS* is the name of the device you want to remove.

2. When you enter the following command, no devices should be displayed. This step verifies that the LUNs have been removed.

   ```
   # lsscsi | grep PATH_SYS
   ```

3. After you remove the LUNS, clean up the device. Enter the following command:

   ```
   # echo "- - -" > /sys/class/scsi_host/host$I/scan
   ```

   where the three dashes refer to the channel, target, and LUN numbers, and *host$i* is the host bus adapter instance. This example cleans up every channel, target, and LUN visible via this host bus adapter instance.

Changing the characteristics of a LUN from the array side

Some arrays provide a way to change the properties of LUNs. In most cases, you must completely stop usage of the device before the device shows the changed characteristics. We recommend that you first take the device offline, change the LUN properties, and then bring back the device online again.

In certain cases, such as EMC BCV and SRDF operations, the device can remain online during this procedure.

In a cluster, perform the steps on all the nodes in the cluster.

To change the properties of a LUN

1. Stop all applications and volumes that are hosted on the device.

   If the device is in use by Veritas Volume Manager (VxVM), perform the following steps:

   For LUNs using Linux LVM over DMP devices, remove the device from the LVM volume group

   ```
   # vgreduce vgname devicepath
   ```

2. Change the LUN characteristics.

3 Bring the device online.

For a Veritas Volume Manager disk:

For LUNs using Linux LVM over DMP devices, add the device back into the LVM volume group

```
# vgreduce vgname devicepath
```

4 Use DMP to perform a device scan.

In a cluster, perform this command on all the nodes.

```
# vxdisk scandisks
```

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.

Storage subsystems contain multiple controllers for redundancy. An online upgrade is done one controller at a time. Dynamic Multi-Pathing (DMP) fails over all I/O to an alternate controller while one of the controllers is undergoing an Online Controller Upgrade. After the controller has completely staged the code, it reboots, resets, and comes online with the new version of the code. The other controller goes through the same process, and I/O fails over to the alternate 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 does not fail 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 does not fail the I/Os for 300 seconds, or until the I/O succeeds.

To verify which arrays support Online Controller Upgrade or NDU, refer to the hardware compatibility list (HCL).

Reformatting NVMe devices manually

You can modify the sector size of NVMe devices by removing the device from VxVM and reformatting it.

To reformat NVMe devices manually

1. Take the disk offline.

   ```
   # vxdisk offline r720xd-114217_intel_nvme0_0
   ```

2. Remove the device from VxVM.

   ```
   # vxdisk rm r720xd-114217_intel_nvme0_0
   ```

3. Removing the NVMe device from the operating system.

   ```
   # echo 1 > /sys/block/nvme0n1/device/device/remove
   ```

4. Refresh the VxVM device tree.

   ```
   # vxdisk scandisks
   ```

5. Verify that device is not present.

   ```
   # vxdisk list | grep nvme
   ```

6. Rescan the NVMe PCI device to add it to the operating system device tree.

   ```
   # echo 1 > /sys/bus/pci/rescan
   # echo 1 > /sys/bus/pci/drivers/nvme/0000\:05\:00.0/rescan
   ```

7 Format the NVMe device to the required sector size using the Intel® SSD Data Center Tool (ISDCT).

```
# isdct start -intelssd 0 -nvmeformat LBAFormat=3 SecureEraseSetting=0 \
ProtectionInformation=0 MetadataSettings=0
WARNING! You have selected to format the drive!
Proceed with the format? (Y|N): y
Formatting...

- Intel SSD DC P3700 Series CVFT5456000V2P0EGN -

Status : NVMeFormat successful.
```

8 Refresh the VxVM device tree.

```
# vxdisk scandisks
```

9 Verify the device.

```
# vxdisk list | grep nvme
r720xd-114217_intel_nvme0_0 auto:none  -   - online invalid
```

Chapter 6

Event monitoring

This chapter includes the following topics:

About the Dynamic Multi-Pathing (DMP) event source daemon (vxesd)

The event source daemon (`vxesd`) is a 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)
  See "Fabric Monitoring and proactive error detection" on page 135.
- Logging of DMP events for troubleshooting (DMP event)
  See "DMP event logging" on page 136.
- Automated device discovery (OS event)
- Discovery of SAN components and HBA-array port connectivity (Fibre Channel and iSCSI)
  See "Dynamic Multi-Pathing (DMP) discovery of iSCSI and SAN Fibre Channel topology" on page 136.

See "Starting and stopping the Dynamic Multi-Pathing (DMP) event source daemon" on page 137.

Fabric Monitoring and proactive error detection

DMP takes a proactive role in detecting errors on paths.

The DMP event source daemon `vxesd` uses the Storage Networking Industry Association (SNIA) HBA API library to receive SAN fabric events from the HBA.

DMP checks devices that are suspect based on the information from the SAN events, even if there is no active I/O. New I/O is directed to healthy paths while DMP verifies the suspect devices.

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 task 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 task 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 task is restarted and the SCSI probes run immediately, without waiting for the next periodic cycle. When the DMP restore task 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 restarts.

To display the current value of the `dmp_monitor_fabric` tunable, use the following command:

```
# vxdmpadm gettune dmp_monitor_fabric
```

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
```

Dynamic Multi-Pathing (DMP) discovery of iSCSI and SAN Fibre Channel topology

The `vxesd` builds a topology of iSCSI and Fibre Channel (FC) 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 the 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

See "About the Dynamic Multi-Pathing (DMP) event source daemon (vxesd)" on page 134.

The event source daemon (`vxesd`) is a Dynamic Multi-Pathing (DMP) component process that receives notifications of any device-related events that are used to take appropriate actions.

DMP notifies `vxesd` of major events, and `vxesd` logs the event in a log file. These events include:

- Marking paths or dmpnodes enabled
- Marking paths or dmpnodes disabled
- Throttling of paths
- I/O error analysis

- HBA and SAN events

You can change the level of detail that is displayed in the system or console log about the DMP events. Use the tunable `dmp_log_level`. Valid values are 1 through 9. The default level is 1.

```
# 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 Dynamic Multi-Pathing (DMP) event source daemon

By default, Dynamic Multi-Pathing (DMP) starts the event source daemon, `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]
```

To view the status of the `vxesd` daemon, use the `vxddladm` utility:

```
# vxddladm status eventsource
```

Handling Fabric Performance Impact Notification (FPIN) events

DMP handles switch level Fabric Performance Impact Notification (FPIN) events to avoid degradation in IO performance. In certain cases, IO performance of the storage subsystem is degraded due to a link failure or a congestion event in the fabric. Multipathing solutions that manage a storage subsystem have the capability to avoid using affected paths and choosing an alternate path for performing IO.

To mitigate such performance degradation scenarios, fabric vendors provide a functionality to receive fabric-specific notifications for each path. To get fabric-specific notifications, the host (end device) has to register with the fabric. After registration, the host receives these events through a Fibre Channel Extended Link Service

(ELS) function that is known as Fabric Performance Impact Notifications (FPINs). Multipathing solutions can use these events to evaluate error conditions and use an alternate path, if required.

How DMP handles FPIN events

The DMP event source daemon (`vxesd`) listens to fabric events through the netlink socket and reacts to the fabric performance notification events. The `vxesd` daemon analyses the FPIN events to identify the affected path and avoids using it until the link or the congestion event subsides. The daemon thus prevents IO throughput degradation. Any performance blips caused due to such events are avoided. After the link or the congestion subsides, The affected path is used for the IO again.

The `vxesd` daemon monitors the FPIN events and marks a path as a standby path on receiving a link integrity event. When a path is marked as a standby, DMP does not send any new IO to that path unless the standby path is the last path to the device. DMP avoids using the standby path to reduce the performance impact due to the event. After the link condition is resolved, DMP moves the standby paths to an active state and resumes IO on that path.

Administering the DMP FPIN monitoring functionality

Note: The FPIN event monitoring functionality does not work with an SRDF or a VPLEX Metro configuration. Veritas recommends that you disable this feature in such configurations.

The DMP tunable `dmp_monitor_fpin_event` tracks the FPIN monitoring functionality status and the tunable is disabled by default.

To display the current status of the FPIN monitoring functionality, use the following command:

```
# vxdmpadm gettune dmp_monitor_fpin_event
```

To enable the FPIN monitoring functionality, use the following command:

```
# vxdmpadm settune dmp_monitor_fpin_event=on
```

To disable the FPIN monitoring functionality, use the following command:

```
# vxdmpadm settune dmp_monitor_fpin_event=off
```

The value of the tunable is persistent across restarts.

Chapter 7

Performance monitoring and tuning

This chapter includes the following topics:

About tuning Dynamic Multi-Pathing (DMP) with templates

Dynamic Multi-Pathing has multiple tunable parameters and attributes that you can configure for optimal performance. DMP provides a template method to update several tunable parameters and attributes with a single operation. The template represents a full or partial DMP configuration, showing the values of the parameters and attributes of the host.

To view and work with the tunable parameters, you can dump the configuration values of the DMP tunable parameters to a file. Edit the parameters and attributes,

if required. Then, load the template file to a host to update all of the values in a single operation.

You can load the configuration file to the same host, or to another similar host. The template method is useful for the following scenarios:

- Configure multiple similar hosts with the optimal performance tuning values. Configure one host for optimal performance. After you have configured the host, dump the tunable parameters and attributes to a template file. You can then load the template file to another host with similar requirements. Veritas recommends that the hosts that use the same configuration template have the same operating system and similar I/O requirements.
- Define multiple specialized templates to handle different I/O load requirements. When the load changes on a host, you can load a different template for the best performance. This strategy is appropriate for predictable, temporary changes in the I/O load. As the system administrator, after you define the system's I/O load behavior, you can customize tuning templates for particular loads. You can then automate the tuning, since there is a single load command that you can use in scripts or cron jobs.

At any time, you can reset the configuration, which reverts the values of the tunable parameters and attributes to the DMP default values.

You can manage the DMP configuration file with the `vxdmpadm config` commands.

See the `vxdmpadm`(1m) man page.

DMP tuning templates

The template mechanism enables you to tune DMP parameters and attributes by dumping the configuration values to a file, or to standard output.

DMP supports tuning the following types of information with template files:

- DMP tunable parameters.
- DMP attributes defined for an enclosure, array name, or array type.
- Veritas naming scheme parameters.

The template file is divided into sections, as follows:

DMP Tunables	Applied to all enclosures and arrays.
Namingscheme	Applied to all enclosures and arrays.
Arraytype	Use to customize array types. Applied to all of the enclosures of the specified array type.

Arrayname	Use if particular arrays need customization; that is, if the tunables vary from those applied for the array type. Attributes in this section are applied to all of the enclosures of the specified array name.
Enclosurename	Applied to the enclosures of the specified Cab serial number and array name. Use if particular enclosures need customization; that is, if the tunables vary from those applied for the array type and array name.

Loading is atomic for the section. DMP loads each section only if all of the attributes in the section are valid. When all sections have been processed, DMP reports the list of errors and warns the user. DMP does not support a partial rollback. DMP verifies the tunables and attributes during the load process. However, Veritas recommends that you check the configuration template file before you attempt to load the file. Make any required corrections until the configuration file validates correctly.

The attributes are given priority in the following order when a template is loaded:

Enclosure Section > Array Name Section > Array Type Section

If all enclosures of the same array type need the same settings, then remove the corresponding array name and enclosure name sections from the template. Define the settings only in the array type section. If some of the enclosures or array names need customized settings, retain the attribute sections for the array names or enclosures. You can remove the entries for the enclosures or the array names if they use the same settings that are defined for the array type.

When you dump a configuration file from a host, that host may contain some arrays which are not visible on the other hosts. When you load the template to a target host that does not include the enclosure, array type, or array name, DMP ignores the sections.

You may not want to apply settings to non-shared arrays or some host-specific arrays on the target hosts. Be sure to define an enclosure section for each of those arrays in the template. When you load the template file to the target host, the enclosure section determines the settings. Otherwise, DMP applies the settings from the respective array name or array type sections.

Example DMP tuning template

This section shows an example of a DMP tuning template.

```
DMP Tunables
        dmp_cache_open=on
        dmp_daemon_count=10
        dmp_delayq_interval=15
        dmp_restore_state=enabled
        dmp_fast_recovery=on
        dmp_health_time=60
        dmp_log_level=1
        dmp_low_impact_probe=on
        dmp_lun_retry_timeout=30
        dmp_path_age=300
        dmp_pathswitch_blks_shift=9
        dmp_probe_idle_lun=on
        dmp_probe_threshold=5
        dmp_restore_cycles=10
        dmp_restore_interval=300
        dmp_restore_policy=check_disabled
        dmp_retry_count=5
        dmp_scsi_timeout=20
        dmp_sfg_threshold=1
        dmp_stat_interval=1
        dmp_monitor_ownership=on
        dmp_monitor_fabric=on
        dmp_native_support=off
Namingscheme
        namingscheme=ebn
        persistence=yes
        lowercase=yes
        use_avid=yes
Arraytype
        arraytype=CLR-A/PF
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arraytype
        arraytype=ALUA
        iopolicy=adaptive
```

```
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arraytype
        arraytype=Disk
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=EMC_CLARiiON
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=EVA4K6K
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=Disk
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Enclosure
        serial=CK200051900278
        arrayname=EMC_CLARiiON
        arraytype=CLR-A/PF
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
```

```
        dmp_lun_retry_timeout=30
Enclosure
        serial=50001FE1500A8F00
        arrayname=EVA4K6K
        arraytype=ALUA
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
Enclosure
        serial=50001FE1500BB690
        arrayname=EVA4K6K
        arraytype=ALUA
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
Enclosure
        serial=DISKS
        arrayname=Disk
        arraytype=Disk
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
```

Tuning a DMP host with a configuration attribute template

You can use a template file to upload a series of changes to the DMP configuration to the same host or to another similar host.

Veritas recommends that you load the DMP template to a host that is similar to the host that was the source of the tunable values.

To configure DMP on a host with a template

1. Dump the contents of the current host configuration to a file.

   ```
   # vxdmpadm config dump file=filename
   ```

2. Edit the file to make any required changes to the tunable parameters in the template.

   The target host may include non-shared arrays or host-specific arrays. To avoid updating these with settings from the array name or array type, define an enclosure section for each of those arrays in the template. When you load the template file to the target host, the enclosure section determines the settings. Otherwise, DMP applies the settings from the respective array name or array type sections.

3. Validate the values of the DMP tunable parameters.

   ```
   # vxdmpadm config check file=filename
   ```

   DMP displays no output if the configuration check is successful. If the file contains errors, DMP displays the errors. Make any required corrections until the configuration file is valid. For example, you may see errors such as the following:

   ```
   VxVM vxdmpadm ERROR V-5-1-0 Template file 'error.file' contains
   following errors:

   Line No: 22  'dmp_daemon_count' can not be set to 0 or less
   Line No: 44  Specified value for 'dmp_health_time' contains
   non-digits
   Line No: 64  Specified value for 'dmp_path_age' is beyond
   the limit of its value
   Line No: 76  'dmp_probe_idle_lun' can be set to either on or off
   Line No: 281 Unknown arraytype
   ```

4. Load the file to the target host.

   ```
   # vxdmpadm config load file=filename
   ```

   During the loading process, DMP validates each section of the template. DMP loads all valid sections. DMP does not load any section that contains errors.

Managing the DMP configuration files

You can display the name of the template file most recently loaded to the host. The information includes the date and time when DMP loaded the template file.

To display the name of the template file that the host currently uses

- `# vxdmpadm config show`

```
TEMPLATE_FILE      DATE              TIME
=============================================
/tmp/myconfig      Feb 09, 2011      11:28:59
```

Resetting the DMP tunable parameters and attributes to the default values

DMP maintains the default values for the DMP tunable parameters and attributes. At any time, you can restore the default values to the host. Any changes that you applied to the host with template files are discarded.

To reset the DMP tunables to the default values

- Use the following command:

```
# vxdmpadm config reset
```

DMP tunable parameters and attributes that are supported for templates

DMP supports tuning the following tunable parameters and attributes with a configuration template.

DMP tunable parameters	See "DMP tunable parameters" on page 147.
DMP attributes defined for an enclosure, array name, or array type.	■ iopolicy ■ partitionsize ■ use_all_paths ■ recoveryoption attributes (retrycount or iotimeout) ■ redundancy ■ dmp_lun_retry_timeout

Naming scheme attributes:

- naming scheme
- persistence
- lowercase
- use_avid

The following tunable parameters are NOT supported with templates:

- OS tunables
- TPD mode
- Failover attributes of enclosures (failovermode)

DMP tunable parameters

DMP provides various parameters that you can use to tune your environment.

Table 7-1 shows the DMP parameters that can be tuned. You can set a tunable parameter online, without a reboot.

Table 7-1 DMP parameters that are tunable

Parameter	Description
`dmp_cache_open`	If this parameter is set to `on`, the first open of a device is cached. This caching enhances the performance of device discovery by minimizing the overhead that is caused by subsequent opens on the device. If this parameter is set to `off`, caching is not performed. The default value is `on`.
`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.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_display_alua_states`	For ALUA arrays, this tunable displays the asymmetric access state instead of PRIMARY or SECONDARY state in the PATH-TYPE[M] column. The asymmetric access state can be: ■ Active/Optimized ■ Active/Non-optimized ■ Standby ■ Unavailable ■ TransitionInProgress ■ Offline The default tunable value is on.
`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, if 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`.
`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.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`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 are critical. 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. 5 or higher — Displays level 1, 2, 3 and 4 messages plus more verbose messages. 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 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
dmp_lun_retry_timeout	Specifies a retry period for handling transient errors that are not handled by the HBA and the SCSI driver. Specify the time in seconds. In general, no such special handling is required. Therefore, the default value of the dmp_lun_retry_timeout tunable parameter is 30. When all paths to a disk fail, DMP fails the I/Os to the application. The paths are checked for connectivity only once. In special cases when DMP needs to handle the transient errors, configure DMP to delay failing the I/Os to the application for a short interval. Set the dmp_lun_retry_timeout tunable parameter to a non-zero value to specify the interval. If all of the paths to the LUN fail and I/Os need to be serviced, then DMP probes the paths every five seconds for the specified interval. If the paths are restored within the interval, DMP detects this and retries the I/Os. DMP does not fail I/Os to a disk with all failed paths until the specified dmp_lun_retry_timeout interval or until the I/O succeeds on one of the paths, whichever happens first.
dmp_monitor_fabric	Determines if DMP should register for HBA events from SNIA HAB APIs. These events improve the failover performance by proactively avoiding the I/O paths that have impending failure. The default setting is off for releases before 5.0 that have been patched to support this DDL feature. The default setting is on for 5.0 and later releases.
dmp_monitor_ownership	Determines whether the ownership monitoring is enabled for ALUA arrays. When this tunable is set to on, DMP polls the devices for LUN ownership changes. The polling interval is specified by the dmp_restore_interval tunable. The default value is on. When the dmp_monitor_ownership tunable is off, DMP does not poll the devices for LUN ownership changes.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`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 Dynamic Multi-Pathing is installed as a component of another Veritas InfoScale product, the default value is `off`. When Dynamic Multi-Pathing is installed as a stand-alone product, 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.
`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 is 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 68.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
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.
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. See "Configuring DMP path restoration policies" on page 82.
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 82.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`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 82.
`dmp_restore_state`	If this parameter is set to `enabled`, it enables the path restoration thread to be started. See "Configuring DMP path restoration policies" on page 82. If this parameter is set to `disabled`, it stops and disables the path restoration thread. If this parameter is set to `stopped`, it stops the path restoration thread until the next device discovery cycle. The default is `enabled`. See "Stopping the DMP path restoration thread" on page 83.
`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 20 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.

Table 7-1 DMP parameters that are tunable *(continued)*

Parameter	Description
`dmp_stat_interval`	The time interval between gathering DMP statistics. The default and minimum value are 1 second.

Appendix A

DMP troubleshooting

This appendix includes the following topics:

Recovering from errors when you exclude or include paths to DMP

You can exclude a path from DMP with the `vxdmpadm exclude` command. You can return a previously excluded path to DMP control with the `vxdmpadm include` command. These commands use the `vxvm.exclude` file to store the excluded paths. The include path and exclude path operations cannot complete successfully if the `vxvm.exclude` file is corrupted.

The following error displays if the `vxvm.exclude` file is corrupted:

```
# vxdmpadm exclude ctlr=c0
VxVM vxdmpadm ERROR V-5-1-3996  File not in correct format
```

DMP saves the corrupted file with the name `vxvm.exclude.corrupt`. DMP creates a new `vxvm.exclude` file. You must manually recover from this situation.

To recover from a corrupted exclude file

1 Reissue the `vxdmpadm include` command or the `vxdmpadm exclude` command that displayed the error.

```
# vxdmpadm exclude ctlr=c0
```

2 View the saved `vxvm.exclude.corrupt` file to find any entries for the excluded paths that are relevant.

```
# cat /etc/vx/vxvm.exclude.corrupt
exclude_all 0
paths
controllers
c4 /pci@1f,4000/pci@4/scsi@4/fp@0,0
```

3 Reissue the `vxdmpadm exclude` command for the paths that you noted in step 2.

```
# vxdmpadm exclude ctlr=c4
```

4 Verify that the excluded paths are in the `vxvm.exclude` file.

```
# cat /etc/vx/vxvm.exclude

exclude_all 0
paths
#
controllers
c0 /pci@1f,4000/scsi@3
c4 /pci@1f,4000/pci@4/scsi@4/fp@0,0
#
product
#

# cat vxvm.exclude

exclude_all 0
paths
#
controllers
c0 c0
c4 c4
#
product
#
```

Downgrading the array support

The array support is available in a single rpm, `VRTSaslapm`, that includes Array Support Libraries (ASLs) and Array Policy Modules (APMs). Each major release of Dynamic Multi-Pathing includes the supported `VRTSaslapm` rpm, which is installed as part of the product installation. Between major releases, Veritas may provide additional array support through updates to the `VRTSaslapm` rpm.

If you have issues with an updated `VRTSaslapm` rpm, Veritas may recommend that you downgrade to a previous version of the ASL/APM rpm. You can only revert to a rpm that is supported for the installed release of Dynamic Multi-Pathing. To perform the downgrade while the system is online, do not remove the installed rpm. Instead,

you can install the previous version of the rpm over the new rpm. This method prevents multiple instances of the `VRTSaslapm` rpm from being installed.

Use the following method to downgrade the `VRTSaslapm` rpm.

To downgrade the ASL/APM rpm while online

- Specify the previous version of the `VRTSaslapm` rpm to the following command:

```
# rpm --force -Uvh VRTSaslapm
```

Appendix B

Reference

This appendix includes the following topics:

Command completion for Veritas commands

Dynamic Multi-Pathing supports command completion for Dynamic Multi-Pathing (DMP) commands.

In this release, command completion is supported only on the bash shell. The shell must be bash version 2.4 or later.

To use this feature, press **Tab** while entering a supported VxVM or DMP command. The command is completed as far as possible. When there is a choice, the command completion displays the next valid options for the command. Enter one of the displayed values. A value in brackets indicates a user-specified value.

Note: Platform-specific options are not supported with command completion.

By default, you can use the command completion feature by invoking the bash shell on every log in. If you want to permanently enable the command completion, use the following command:

```
# vxdctl cmdcompletion enable
```

The enable command completion creates the `.bash_profile` file, if it is not present.

To permanently disable the command completion, use the following command:

```
# vxdctl cmdcompletion disable
```

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

The following commands support command completion:

- vxddladm
- vxdisk
- vxdmpadm