SPARCstorage Array Man Pages

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043 U.S.A.
415 960-1300 FAX 415 969-9131

Part No: 801-7367-12
Rev A of March 1995

NAME

ssaadm – Administration program for the SPARCstorage Array

SYNOPSIS

/usr/sbin/ssaadm [**–v**] *subcommand* [*option* . . .] *pathname* . . .

DESCRIPTION

The **ssaadm** program is an administrative command that manages the SPARCstorage Array. **ssaadm** performs a variety of control and query tasks depending on the command line arguments and options used.

The command line must contain a subcommand and at least one pathname. It may also contain options and other parameters depending on the subcommand. The subcommand is applied to each of the pathnames on the command line.

pathname specifies the SPARCstorage Array controller or a disk in the SPARCstorage Array. The controller name is specified by its physical name, e.g.,

/devices/.../.../SUNW,soc@3,0/SUNW,pln@axxxxxxx,xxxxxxxx:ctlr,

or by a name of the form **c***N*, where *N* is the logical controller number. **ssaadm** uses the **c***N* name to find an entry in the **/dev/rdsk** directory of a disk that is attached to the SPARCstorage Array controller. The **/dev/rdsk** entry is then used to determine the physical name of the SPARCstorage Array controller. A disk in the SPARCstorage Array is specified by its logical or physical device name, e.g., **/dev/rdsk/c1t0d0s2** or **/devices/.../.../SUNW,soc@3,0/SUNW,pln@axxxxxxx,xxxxxxxx/ssd@0,0:c,raw**. See **disks**(1M) for more information on logical names for disks and controllers.

OPTIONS

–v verbose mode.

Other options specific to each subcommand are described below.

USAGE

Subcommands

auto_reallocate [**–s**] **–e** *pathname*

auto_reallocate [**–s**] **–d** *pathname*

Enable or disable the automatic reallocation of bad blocks. This subcommand can be issued only to the SPARCstorage Array controller.

–s This option causes the SPARCstorage Array to save the change so it will persist across power-cycles.

–e Enable the automatic reallocation of bad blocks.

–d Disable the automatic reallocation of bad blocks.

display [**–p**] *pathname*

Display configuration information for the specified unit(s) or display performance information for the specified SPARCstorage Array controller. If *pathname* specifies the controller, the configuration information is displayed for all disks in the SPARCstorage Array.

–p Display performance information for the specified SPARCstorage Array controller. The accumulation of the performance statistics must be enabled using the **perf_statistics**

subcommand before displaying the performance information. If it is not, all of the I/O's per second will be displayed as zeros.

The performance display reports the following information: BUSY shows how busy the controller in the SPARCstorage Array is, expressed as a percentage. IOPS shows the total I/O's per second for the SPARCstorage Array. The entries for each disk show the total number of I/O's per second.

download –**f** *filename pathname*

Download the prom image specified by *filename* to the SPARCstorage Array controller FEPROMs. When the download is complete the SPARCstorage Array must be reset in order to use the downloaded code. **NOTE:** The download modifies the FEPROM on the SPARCstorage Array and should be used with **CAUTION**.

fast_write [–**s**] –**c** *pathname*

fast_write [–**s**] –**d** *pathname*

fast_write [–**s**] –**e** *pathname*

Enable or disable the use of the NVRAM to enhance the performance of writes in the SPARCstorage Array. *pathname* may refer to the SPARCstorage Array controller or to an individual disk.

–**s** This option causes the SPARCstorage Array to save the change so it will persist across power-cycles.

–**c** Enable fast writes for synchronous writes only.

–**e** Enable fast writes.

–**d** Disable fast writes.

perf_statistics –**e** *pathname*

perf_statistics –**d** *pathname*

Enable or disable the accumulation of performance statistics for the specified SPARCstorage Array controller. The accumulation of performance statistics must be enabled before using the **display** –**p** subcommand. This subcommand can be issued only to the SPARCstorage Array controller.

–**e** Enable the accumulation of performance statistics.

–**d** Disable the accumulation of performance statistics.

purge *pathname*

Purge any fast write data from NVRAM for one disk, or all disks if the controller is specified. This option should be used with caution, usually only when a drive has failed.

reserve *pathname*
Reserve the specified controller(s) or disk(s) for exclusive use by the issuing host.

release *pathname*
Release a reservation held on the specified controller(s) or disk(s).

start [**–t** *tray-number*] *pathname*
Spin up the specified disk(s). If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

–t Spin up all disks in the tray specified by *tray-number*. *pathname* must be of the controller.

stop [**–t** *tray-number*] *pathname*
Spin down the specified disk(s). If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

–t Spin down all disks in the tray specified by *tray-number*. *pathname* must be of the controller.

sync_cache *pathname*
Flush all outstanding writes for the specified disk(s) from NVRAM to the media. If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

SEE ALSO **disks**(1M)

NOTES For more information, see the *SPARCstorage Array Users Guide*.

NAME

vxassist – create, mirror, backup, grow, shrink, and move volumes

SYNOPSIS

vxassist [*options*] [**–b**] **make** *volume length* [*attribute* ...]
vxassist [*options*] [**–b**] **mirror** *volume* [*attribute* ...]
vxassist [*options*] [**–b**] **move** *volume* **!***storage-spec* ... [*attribute* ...]
vxassist [*options*] [**–b**] **growto** *volume newlength* [*attribute* ...]
vxassist [*options*] [**–b**] **growby** *volume lengthchange* [*attribute* ...]
vxassist [*options*] **shrinkto** *volume newlength*
vxassist [*options*] **shrinkby** *volume lengthchange*
vxassist [*options*] [**–b**] **snapstart** *volume* [*attribute* ...]
vxassist [*options*] **snapshot** *volume newvolume*
vxassist [*options*] **snapwait** *volume*

DESCRIPTION

The **vxassist** utility is an interface to the VERITAS Volume Manager that finds space for and creates volumes, adds mirrors and logs to existing volumes, extends and shrinks existing volumes, provides for the migration of data from a specified set of disks, and provides facilities for the on-line backup of existing volumes.

The command is supplied a keyword that selects the action to perform.

Each operation can be applied to only one disk group at a time, due to internal implementation constraints. For the **make** operation, attributes that name specific disks may be used to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). For other operations, the *volume* operand is used. A specific disk group can be forced with **–g** *diskgroup*. With no disk attributes are specified, the **make** operation defaults to using the **rootdg** disk group. An alternate default disk group can be specified in a defaults file (**/etc/default/vxassist**).

Many operations can take a set of attributes that specify how volumes should be layed out, which sets of disks they should be build on, amount other things. Attributes are of two basic types: storage specifications, and attribute settings. Storage specification attributes are either simple disk names (e.g., **disk01**), or they can select groups of disks (e.g., **ctlr:c1** specifies all disks on controller 1). Excluded (negated) storage specification can be specified with a prefix of **!**. For example, the arguments **ctlr:c1** and **!target:c1t5** can be combined to indicate that storage should be allocated from controller 1, but not from SCSI target 5 on that controller.

Other attributes are of the form *attrname***=***value*; these additional attributes can specify the type of a volume (mirrored, RAID-5, striped, logged), layout policies (contiguous, spanning), mirroring requirements (mirror across controllers or SCSI targets), constraint parameters (constrain allocations to a single controller), and more. A complete list of attribute specifications is given later in this man page.

If no non-excluded storage specification attributes are given, then any non-reserved, non-volatile, non-spare, non-excluded disk may be used to allocate storage. Attributes may constrain the selection, particularly with respect to the selection of multiple disks. For example, the command:

vxassist make mirvol 500m layout=mirror,log mirror=ctlr !ctlr:c2

requests that **vxassist** create a new mirrored volume on any disks that are not on controller 2. The selection of disks is constrained by the **mirror=ctlr** attribute such that no disks within mirror can be on the same controller as any disks on the other mirror.

The recognized operation keywords are:

make Create a volume with the specified name and the specified length. The length is specified as a standard Volume Manager length [see **vxintro**(1M)]. Attributes can be specified to indicate various desired properties and storage locations for the created volume.

mirror Create a new mirror (or plex) and attach it to the volume. This operation is allowed only if the volume is enabled. Attributes can be specified to indicate various desired properties and storage locations for the created volume. Attributes that constrain mirroring (such as requiring that mirrors be on separate groups of disks) apply between the existing mirrors of the volume and the new mirror.

move Move subdisks within the named volume off of the excluded storage specified on the command line. Excluded storage is specified with a prefix of **!**. The move operation requires that at least one excluded storage specification be given.

If the volume is enabled, then subdisks within detached or disabled plexes (or detached log or RAID-5 subdisks) will be moved without recovery of data. If the volume is not enabled, then stale log or RAID-5 subdisks, or subdisks within STALE or OFFLINE plexes, will be moved without recovery; if there are other subdisks within a non-enabled volume that require moving, then the **move** operation will fail.

For enabled subdisks, in enabled plexes within an enabled volume, the data within subdisks will be moved to the new location, without loss of availability (or redundancy) of the volume.

growto and **growby**
Increase the length of the named volume to the length specified by *newlength* (**growto**), or by the length specified by *lengthchange* (**growby**). The new length or change in length are specified as standard Volume Manager lengths [see **vxintro**(1M)]. The **growto** operation fails if the new length is not greater than the current volume length. The length of the volume is increased by extending existing subdisks in the volume, or by adding and associating new subdisks. Plexes that are not fully populated already (i.e., that are *sparse*) are left unchanged. Also, log-only plexes are left unchanged.

Attributes can be specified to indicate various desired properties for the new allocations. Any mirroring constraints will still apply between all extensions to the existing mirrors and the other mirrors. Growing of a volume requires that the volume be enabled.

No attempt will be made, by **vxassist** to adapt the file system that resides on the volume (or another other users of the volume). It is expected that any necessary

application notifications will be made *after* the grow operation completes successfully.

shrinkto and **shrinkby**
Decrease the length of the named volume to the length specified by *newlength* (**shrinkto**), or by the length specified by *lengthchange* (**shrinkby**). The new length or change in length are specified as standard Volume Manager lengths [see **vxintro**(1M)]. The **shrinkto** operation fails if the new length is not less than the current volume length.

The length of a volume is decreased by removing and shortening subdisks to leave each plex with the desired volume length. The freed space can then be allocated for use by other volumes. Log-only plexes are left unchanged.

No attempt will be made, by **vxassist** to adapt the file system that resides on the volume (or another other users of the volume). It is expected that any necessary application notifications will be made *before* the shrink operation is initiated.

addlog Add a log to a mirrored or RAID-5 volume. A log can be added to a mirrored volume only if the logging type of the volume is **DRL** (this is the default logging type). When adding the first log to a mirrored or RAID-5 volume, the **logen** attribute can be used to specify the length of the log; otherwise, the **loglen** attribute will be ignored. Other attributes can be specified to indicate desired properties for the new allocations. Mirroring constraints do not apply to logs, though storage constraints can be specified that constrain the logs to disk sets already used by the volume.

Adding a log to a DRL mirrored volume involves creating a single log subdisk and a new plex to contain that subdisk. The new plex will then be attached to the volume. Adding a log to a RAID-5 volume involves creating a new plex that will be attached to the volume as a log plex.

snapstart and **snapshot**
Create a temporary mirror and attach it to the named volume. When the attach completes, the mirror will be considered a candidate for selection by the **snapshot** operation. The **snapshot** operation takes one of these attached temporary mirrors and creates a new volume with the temporary mirror as its one plex. Attributes can be specified to indicate desired properties of the snapshot mirror.

Some usage types will attempt to synchronize any in-memory data associated with the volume (such as unwritten file system modifications) when the **snapshot** operation is done. In particular, if the **fsgen** usage type is used with a volume containing a VxFS file system, then cooperating procedures ensure that all file system data is consistently flushed to the volume. For the **ufs** and **s5**, the synchronization operation consists of a call to **sync**(1M), which will make the snapshot a better image, but which may leave some inconsistencies between in-memory file system data and the data residing on the backup image.

snapwait
If a **snapstart** mirror attach is done as a background task (such as using the **–b**

option), it may be convenient to wait for an attached mirror to become available. The **snapwait** operation waits for such an attach to complete on the named volume. When a snapshot attach has completed, the operation exits.

OPTIONS

The following options are recognized:

–g *diskgroup*
: Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *medianame* operands (if any) for the **vxassist make** operation, or based on the *volume* operands for all other operations.

–U *usetype*
: Limit the operation to apply to this usage type. Attempts to affect volumes with a different usage type will fail. For a **vxassist make** operation, this indicates the usage type to use for the created volume. Otherwise, the default is used and is determined by the existence of an entry in the **/etc/default/vxassist** file or else is set to the **fsgen** usage type.

–o *useopt*
: Pass in usage-type-specific options to the operation. A certain set of operations are expected to be implemented by all usage types:

  slow[**=***iodelay*]
  : Reduce the system performance impact of copy operations. Copy and plex consistency recovery operations are usually a set of short operations on small regions of the volume (normally from 16 kilobytes to 128 kilobytes). This option inserts a delay between the recovery of each such region. A specific delay can be specified with *iodelay* as a number of milliseconds, or a default is chosen (normally 250 milliseconds).

  iosize=*size*
  : Perform copy and recovery operations in regions with the length specified by *size*, which is a standard Volume Manager length number [see **vxintro**(1M)]. Specifying a larger number typically causes the operation to complete sooner, but with greater impact on other processes using the volume. The default I/O size is typically between 32 and 128 kilobytes.

–b
: Perform extended operations in background. This applies to plex consistency recovery operations for **vxassist make**, **growto**, and **growby**. This flag also applies to plex attach operations started by either **vxassist mirror** and **vxassist snapstart**.

–d *defaults*
: Specify a file containing defaults for various attributes related to volume creation and space allocation. If not specified, this defaults to **/etc/default/vxassist**.

ATTRIBUTES

Attribute values for various purposes can be specified with arguments of the form *attribute***=***value*. Attributes can also be passed in through a defaults file. Default attribute values can be stored in the file **/etc/default/vxassist**.

Attributes are selected according to the order in which they are scanned. In general they are taken in decreasing priority of being specified on:

1. The command line.
2. The user's defaults file. As supplied with the **-d** command line argument.
3. The system defaults file. As specified in **/etc/default/vxassist**.

Attributes from all sources have the same form. However, in some cases, command-line attributes change default behaviors in ways that defaults-file supplied attributes do not. In particular, references to mirroring (such as specifying a mirror count) or logging (such as specifying a log count or length) on the command line will cause mirroring or logging to happen by default. If such attributes are specified in a defaults file, then they just indicate the attributes that would be used if mirroring or logging were enabled.

Attributes are either storage specifications (possibly negated), or are in the form *attribute***=***value*. In a defaults file, attributes are whitespace separated or are on separate lines. Blank lines in a defaults file are ignored, and comments can be included with the standard **#** convention.

Storage Specifications

Storage specification attributes have one of the following forms:

[**!**]*diskname*
: Specify (or exclude) the given disk. *diskname* refers to a disk media record name in a volume manager disk group.

[**!**]*diskclass***:***instance*
: Specify (or exclude) a set of disks. *diskclass* names a particular type of disk grouping (for example, **ctlr** to indicate groups of disks on a single controller), and *instance* specifies which grouping of that type (for example **c1**). Each type of disk class has a particular format for specifying instances of the class.

*diskclass***:same**
: Specify that the allocation must be constrained to include disks from the same instance of the given class. For example, a storage specification of **diskparam:same** indicates that all disks selected must have the same basic parameters (tracksize, cylinder size, size, and revolutions-per-minute), but does not specify which specific parameters to use.

The defined disk classes (and alternate names, aliases, for those classes) are as follows:

ctlr (alias **c**)
: Specify disks from a particular controller. The controller is identified in the form **c***number*, where *number* is a decimal controller number starting at 0.

ctype (alias **ctlrtype**)
: Specify disks from a particular type of controller. The controller type is usually a driver name. For example, on Suns, the regular SCSI controller type can be identified as **ctype:esp**. As a special case, the SPARCstorage Array controller type is specified as **ctype:ssa**.

da
: Specify a specific volume manager disk by Disk Access Record name. An example of a disk access record is **c1t5d0s6**, which indicates a special volume manager

disk defined on slice 6 of disk **c1t5d0**.

device (alias **d**)
Specify all volume manager disks on a specific physical disk. The physical disk is specified in the form **c***controller***t***target***d***disk*. Normally, only one volume manager disk device is created for each physical disk. However, the **vxdisk define** operation [see **vxdisk**(1M)] can be used to create additional volume manager disk devices on selected partitions.

diskgeom (alias **geom**)
Specify disks with a particular geometry. The geometry is specified in the form *cyls***.**heads**.**sectors, to indicate the number of cylinders, heads, and sectors per track, of the disk. Many modern drives have variable geometries (or geometries that don't fit the standard conventions). For such drives, the geometry parameters used are whatever the drive reports.

diskparam
Specify disks with particular parameters. The parameters are specified in the form *cyls***.**heads**.**sectors*.rpm*, to indicate cylinders, heads, sectors per track, and the number of revolutions per minute of the drive. As with the **diskgeom** parameter, the instances of the **diskparam** disk class depend upon the values reported by the drive, and may not be accurate.

diskrm (alias **rpm**)
Specify disks with a particular number of revolutions per minute. The revolutions per minute is given as a simple decimal number.

dm (alias **disk**)
Specify a volume manager disk by Disk Media Record name, in the disk group of the volume. Specifying a disk class type of **dm** or **disk** is equivalent to giving a storage specification with no disk class name.

target (alias **1**)
Specify disks on the same SCSI target address in the same controller. The target is specified in the form **c***controller***t***target*. For example, **t:c2t4** selects disks on target 4 of controller 2.

tray Specify disks in the same removable tray (for SPARCstorage Array controllers). Trays in the SPARCstorage Array are grouped as two SCSI targets per tray (e.g., targets 0 and 1 are in the same tray). Trays are specified in the form **c***controller***tray***tray*, where *controller* identifies a particular SPARCstorage Array controller and *tray* is the number of a tray (0 for SCSI targets 0 and 1, 1 for targets 2 and 3, or 2 for targets 4 and 5).

The **diskgeom** and **diskparam** attributes are most often useful in the form **diskgeom**:**same** and **diskparam**:**same**, to indicate that all disks used for creating a volume should be on similar disk drives.

Other Attributes

Other attributes are of the form *attribute***=***value*. The attributes of this form are. The *attribute* name in an attribute value pair will never contain a colon, so it is possible to specify a disk that has an equal sign in its name using the storage specification **dm**:**disk01=a**.

Without the **dm:** prefix, **disk01=a** would yield an error indicating an unrecognized attribute.

Defined attributes (and common aliases) are:

alloc=*storage-spec*[,*storage-spec*,...]
This is provided as an alternate syntax for specifying storage as single attributes. It is mostly interesting in a defaults file, so that all attributes (including storage specifications) will be in the *attribute*=*value* format. Any number of storage specifications can be specified, separated by commas. More than one **alloc** attribute can be specified, in which case they are logically concatenated to form one longer list.

comment=*comment*
Specify a comment to give to a volume when it is created. This comment can be displayed with **vxprint -l**, and can be changed, at a later time, with **vxedit set**. This attribute is used only with the **make** operation.

diskgroup=*disk-group* (alias: **dg**)
Specify the disk group for an operation. If a disk group is specified in a defaults file, then it just specifies the default disk group to use for the **make** operation, if no other means of determining the disk group can be used. If specified as a command line attribute, it has the same effect as specifying a disk group to the **–g** option (the operation is forced to apply to the given disk group).

fstype=*file-system-type*
Specify the file system type for a new volume (this is used only with the **make** operation). The file system type for a volume is usually determined when needed by running the **fstyp** utility [see **fstyp**(1M)], but can be specified explicitly. The file system type parameter is used the with volume manager **vxresize** utility and, as well as with volume snapshots, copies, and dissociates (to select features that are used with the VERITAS VxFS file system).

The file system type can be changed at any later time with the **vxedit set** operation.

group=*owning-group*
Set the owning group for a new volume. The group can be specified numerically or with a system group name. This attribute is used only with the **make** operation. By default, volumes are created in group 0.

init=*initialization-type*
Specify the means for initializing a new volume. The default method (which can be selected explicitly with **init=default**) is to call **vxvol start** to do a usage-type-specific default initialization operation. A new volume can be left unintialized with **init=none**. The most useful non-default initialization that can be specified is **init=zero**, to clear the volume before enabling it for general use.

layout=*layout-spec*[,*layout-spec*,...]
Specify a volume or plex layout type (RAID-5, mirrored, unmirrored, striped, cylinder alignment, etc), and turn on or off some features (logging). The list of layout specifications is given in the next section. By default, unmirrored, non-

striped volumes are created with no log.

loglen=*length*

Specify a log length to use for dirty-region log subdisks for mirrored volumes or for RAID-5 log plexes. If a log length is specified on the command line, then logging will be enabled by default.

logtype=drl | **region** | **none**

Specify the logging type for a mirrored volume. Dirty-region-logging (the default) can be selected with either **logtype=drl** or **logtype=region**. Logging can be disabled altogether with **logtype=none**.

max_nraid5column=*number* (alias: **maxraid5columns**)

Specify the default maximum number of stripe columns for a RAID-5 volume (default value: 8). The rules for selecting the default number of RAID-5 stripe columns is desribed later.

max_ncolumn=*number* (alias: **maxcolumns**)

Specify the default maximum number of stripe columns, either for a RAID-5 volume (if **max_nraid5stripe** is not also specified) or for a striped plex (default value: 8). The rules for selecting the default number of stripe columns is described later.

max_regionloglen=*length* (alias: **maxregionloglen**)

Specify the maximum default dirty-region-logging subdisk log length. If the user does not specify the dirty-region log length for a volume, when creating the first log for a mirrored volume, **vxassist** uses a simple formula based on the log length. The default length will not be bounded by **max_regionloglen** (default value: 32k).

min_nraid5column=*number* (alias: **minraid5columns**)

Specify the default minimum number of stripe columns for a RAID-5 volume (default value: 3). The policy for selecting a default number of RAID-5 stripe columns will not be allowed to create a RAID-5 volume with fewer than **min_nraid5stripe** stripe columns.

min_nstripe=*number* (alias: **minstripes**)

Specify the default minimum number of stripe columns for either a RAID-5 volume (if **min_nraid5stripe** is not also specified) or for a striped plex (default value: 2). The policy for selecting a default number of stripe columns will not be allowed to select fewer than this number of columns.

mirror=*number* | **yes** | **no** | *diskclass*[,...]

Specify various mirroring parameters. Multiple mirroring parameters can be given, each separated by a comma. A decimal number indicates a specific number of mirrors to create, when creating a mirrored volume (equivalent to **nmirror=***number*). **yes** indicates that volumes should be mirrored by default (equivalent to **layout=mirror**). **no** indicates that volumes should be unmirrored by default (equivalent to **layout=nomirror**).

Any other mirroring parameters specify that volumes should be mirrored across instances of a particular disk class. For example, **mirror=target** specifies that

volumes should be mirrored between SCSI target addresses. Each mirror can contain disks from any number of instances of the disk class, but different mirrors cannot use disks from the same instance of the disk class.

mode=*permissions*
Specify the permissions for the block and character device nodes created for a new volume. The mode can be specified either as an octal number or symbolically. A symbolic mode is specified using the syntax given in **chmod**(1M). This attribute is used only with the **make** operation. The default mode for a volume gives read and write access only to the owner.

nlog=*number*
Specify the number of logs to create, by default, for a RAID-5 or mirrored volume (presuming that logs will be created). The number of logs to create can be specified independently for RAID-5 or mirrored volumes using the **nraid5log** and **nregionlog** attributes.

nmirror=*number*
Specify the number of mirrors to create when mirroring a volume (default value: 2).

nraid5log=*number*
Specify the number of log plexes to create for a new RAID-5 volume (default value: 1). This attribute is used only with the **make** operation.

nraid5stripe=*number* (alias: **raid5stripes**)
Specify the number of stripe columns to use when creating a new RAID-5 volume (the default is to adjust the number to available disks). This attribute is used only with the **make** operation.

nregionlog=*number*
Specify the number of log subdisks to create for a new mirrored volume (default value 1). This attribute is used only with the **make** operation, and only if logging is requested for the volume.

nstripe=*number* (alias: **stripes**, **ncolumn**)
Specify the number of stripe columns to use when creating a new RAID-5 volume (with the **make** operation) or when creating a striped plex (with the **make**, **mirror**, and *snapstart* operations). The default is to adjust to the number of available disks.

raid5_stripeunit=*width* (alias: **raid5_stwid**)
Specify the stripe unit width to use when creating a new RAID-5 volume (default value: 16k). This attribute is used only with the **make** operation.

raid5loglen=*length*
Specify the log length to use when adding the first log to a RAID-5 volume. The default is four times the full stripe width (the stripe unit width times the number of stripe columns).

regionloglen=*length*
Specify the log subdisk length to use when adding the first log subdisk to a

mirrored volume. The default is chosen based on a formula involving the volume length.

stripe_stripeunit=*width* (alias: **stripe_stwid**)
Specify the stripe unit width to use when creating striped plexes to attach to a volume. When attaching a new volume, the default is to use the same stripe width as any other striped plexes in the volume; otherwise. If the volume does not yet contain striped plexes, the default value is 64k.

stripeunit=*width* (alias: **stwid**,
stripewidth)" Specify the stripe unit width to use for either a RAID-5 volume or for striped plexes.

user=*owning-user*
Specify the owning user for a new volume (default value: **root**). The user can be specified either numerically or it can be a system login name. This attribute is used only with the **make** operation.

usetype=*volume-usage-type*
Specify the usage type to use when creating a new volume (default value: **raid5** for RAID-5 volumes; **fsgen**, otherwise). This attribute is used only with the **make** operation. The usage type can also be specified in the option list with **–U**.

wantalloc=*storage-spec*[,*storage-spec*,...]
Specify a set of desired storage specifications. This is useful in a defaults file to indicate desired storage specifications that should be discarded if they fail to yield a reasonable set of allocations. The format is the same as for the **alloc** attribute (a comma-separated list of storage specifications).

For example, a defaults file can name a specific controller type to use for allocations, if possible. As soon as all disks on that type of controller are full, other controllers will be used.

wantmirror=*diskclass*[,*diskclass*,...]
Specify a desired list of disk class mirroring specifications. This is useful in a defaults file to indicate a set of desired mirroring constraints that can be dropped if they fail to yield a reasonable set of allocations.

For example, a defaults can specify that volumes should be mirrored between removable trays of disks, if possible. Then, if the set of volumes builds up in such a way that a new volume cannot be mirrored across pluto trays, then this constraint will be dropped.

Layout Specifications

The **layout** attribute specifies a comma-separated list of simple parameters (with no arguments) that apply to **vxassist** operations.

Defined layout specifications are:

mirror, **nomirror**, **raid5**
New volumes should be mirrored, unmirrored (default), or RAID-5, respectively.

stripe, **nostripe**
New plexes should be striped or unstriped, respectively. When creating a new

volume, the default is **nostripe**. When adding a new plex to an existing volume, the default is **stripe**, if the volume already has one or more striped plexes, and is **nostripe**, otherwise.

span, **nospan**
Allow (default) or disallow plexes, regular stripe columns, or RAID-5 stripe columns from spanning multiple disks. If **nospan** is indicated, then plexes or columns can be formed from multiple regions of the same disk, but cannot be formed from more than one disk.

contig, **nocontig**
Disallow or allow (default) plexes, regular stripe columns, or RAID-5 stripe columnes from using multiple regions of disk. If **contig** is specified, then plexes and columns must be allocated from a single contiguous region of disk. If this is not possible, the allocation fails. By default, **vxassist** already tries to allocate space contiguously but will use multiple regions or multiple disks if needed.

log, **nolog**
Create (or don't create) dirty-region logs (for mirrored volumes) or log plexes (for RAID-5 volumes) when creating a new volume. This attribute can be specified independently for mirrored and RAID-5 volumes with the **raid5log** and **regionlog** layout specifications.

raid5log, **noraid5log**
Create (default) or don't create log plexes for RAID-5 volumes.

regionlog, **noregionlog**
Create or don't create (default) dirty-region log subdisks for mirrored volumes.

diskalign, **nodiskalign**
Align (default) or don't align subdisks on cylinder boundaries. When alignment is not disabled, subdisks are are created beginning on cylinder boundaries and are extended to match the end of a cylinder. Dirty-region log subdisks, however, are not cylinder aligned (they are usually only 2 or 3 blocks long). Instead, when creating log subdisks, spaces are located from the available disks that could not be turned into regular subdisks because the spaces aren't cylinder aligned. For example, once one cylinder is used for a log subdisk, that cylinder could not be used to create an aligned data (or RAID-5 log) subdisk, so other log subdisks will be created there until that cylinder fills up.

FILES

/etc/default/vxassist
System default settings for **vxassist** attributes.

EXIT CODES

The **vxassist** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility.

See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

vxintro(1M), **vxedit**(1M), **vxmake**(1M), **vxmend**(1M), **vxplex**(1M), **vxresize**(1M), **vxsd**(1M), **vxvol**(1M).

NAME

vxbootsetup – setup system boot information on a Volume Manager disk

SYNOPSIS

/etc/vx/bin/vxbootsetup [*medianame* ...]

DESCRIPTION

The **vxbootsetup** utility configures physical disks so that they can be used to boot the system. Before **vxbootsetup** is called to configure a disk, mirrors of the root, swap, **/usr** and **/var** volumes (if they exist) should be created on the disk. These mirrors should be restricted mirrors of the volume. The **vxbootsetup** utility configures a disk by writing a boot track at the beginning of the disk and by creating physical disk partitions in the UNIX VTOC that match the mirrors of the root, swap, **/usr** and **/var**.

With no **medianame** arguments, all disks that contain usable mirrors of the root, swap, **/usr** and **/var** volumes are configured to be bootable. If **medianame** arguments are specified, only the named disks are configured.

vxbootsetup requires that the root volume be named **rootvol** and to have a usage type of **root**. The swap volume is required to be named **swapvol** and to have a usage type of **swap**. The volumes containing **/usr** and **/var** (if any) are expected to be named **usr** and **var**, respectively.

Partitions which overlay restricted mirrors of the root volume are assigned a partition number of 0. Partitions created to overlay restricted mirrors of the swap volume are assigned a partition number of 1. Mirrors of the volumes containing **/usr** and **/var** are assigned one of the free partition numbers on the disk. All created partitions are given partition types appropriate for use with the system boot process.

Root, swap, **/usr** and **/var** volumes are created when the original system boot disk is encapsulated with the **vxencap** utility or when the Volume Manager is first setup on the system with **vxinstall**. (**/usr** and **/var** are only encapsulated if they existed on the system; see the VERITAS Volume Manager (VxVM) Installation Guide and the **vxencap**(1M) man page for details on encapsulating the boot disk.

See the appendix on recovery in the *VERITAS Volume Manager (VxVM) System Administrator's Guide* for detailed information on how the system boots and how the Volume Manager impacts on the system boot process.

The following utilities call **vxbootsetup** automatically: **vxmirror**, **vxrootmir**, **vxresize**, and **vxencap**. If you use **vxassist** or **vxmake** and **vxplex** to create mirrors of the root, swap and **/stand** volumes on a disk, you must call **vxbootsetup** directly to make that disk bootable.

SEE ALSO

vxintro(1M), **vxassist**(1M), **vxevac**(1M), **vxinstall**(1M), **vxmake**(1M), **vxmirror**(1M), **vxplex**(1M), **vxrootmir**(1M), **vxresize**(1M), **disksetup(1M), edvtoc**(1M).
Also, see the appendix on recovery in the *VERITAS Volume Manager (VxVM) System Administrator's Guide* and the Volume Manager installation guide.

NAME

vxconfigd – Volume Manager configuration daemon

SYNOPSIS

vxconfigd [**–kfd**] [**–r reset**] [**–m** *mode*] [**–x** *debug*] [**–D** *diag_portal*] [**–R** *request_portal*]

DESCRIPTION

The Volume Manager configuration daemon, **vxconfigd**, is responsible for maintaining configurations of disks and disk groups in the VERITAS Volume Manager. **vxconfigd** takes requests from other utilities for configuration changes, and communicates those changes to the kernel and modifies configuration information stored on disk. **vxconfigd** is also responsible for initializing the Volume Manager when the system is booted.

The **vxconfigd** utility has the following options:

–k If a **vxconfigd** process is running already, then kill it before any other startup processing. This is useful for recovering from a hung **vxconfigd** process. Killing the old **vxconfigd** and starting a new one should not cause any problems for volume or plex devices that are being used by applications or that contain mounted file systems.

–f Start **vxconfigd** in the foreground. This is often useful when debugging **vxconfigd**, or when tracing configuration changes. If this flag is not used, **vxconfigd** will fork a background daemon process. The foreground process will exit as soon as **vxconfigd** startup processing completes.

–d This is equivalent to **–m disable**, which starts **vxconfigd** in disabled mode.

–r reset Reset all Volume Manager configuration information stored in the kernel as part of startup processing. This will fail if any volume or plex devices are currently in use. This option is primarily useful for testing or debugging.

–m *mode* Set the initial operating mode for **vxconfigd**. Possible values for *mode* are:

enable Start fully enabled (default). This will use the **volboot** file to bootstrap and load in the **rootdg** disk group. It will then scan all known disks looking for disk groups to import, and will import those disk groups. This will also set up the **/dev/vx/dsk** and **/dev/vx/rdsk** directories to define all of the accessible volume manager devices. If the **volboot** file cannot be read or if the **rootdg** disk group cannot be imported, **vxconfigd** will be started in *disabled* mode.

disable
Start in disabled mode. This creates a rendezvous file for utilities that perform various diagnostic or initialization operations. This can be used with the **–r reset** option as part of a command sequence to completely reinitialize the Volume Manager configuration. Use the **vxdctl enable** operation to enable **vxconfigd**.

boot Handle boot-time startup of the volume manager. This starts the **rootdg** disk group and the root and **/usr** file system volumes. This mode is capable of operating before the root file system is remounted to read-write. **vxdctl enable** should be called later in the boot

sequence to trigger **vxconfigd** to rebuild the **/dev/vx/dsk** and **/dev/vx/rdsk** directories.

–x *debug* Turn on various parameters used for debugging or other miscellaneous aspects of **vxconfigd** operation. The *debug* option argument is a decimal number, which will set a tracing output level, or one of the following strings:

timestamp or **mstimestamp**
Attach a date and time-of-day timestamp to all messages written by **vxconfigd** onto the console. If **mstimestamp** is used, then a millisecond value is also displayed, allowing detailed timing of **vxconfigd**'s operation.

syslog or **nosyslog**
These options are provided only on systems where **vxconfigd** is compiled with support for the *syslog*( ) library calls [see **syslog**(3) and **syslogd**(1M)]. **vxconfigd** can support use of *syslog*( ) to log all of its regular console messages. This support is either available by default (in which case **–x nosyslog** is required to disable its use), or it can support it as a run-time option (in which case **–x syslog** can be used to enable its use).

For Solaris, **–x syslog** must be supplied to **vxconfigd** to enable use of *syslog*( ). The system can be changed to use this option at boot-time by editing the VxVM startup scripts.

NOTE: If the **syslog** option is enabled, then all console output will be directed through the *syslog*( ) interface. However, both **syslog** and **log** (described below) can be used together to get reliable logging, into a private log file, along with distributed logging through **syslogd**.

log or **nolog**
As an alternative to the use of *syslog*( ), **vxconfigd** can directly log all of its console output to a file. This logging is reliable, in that any messages which are output just before a system crash will be available in the log file, presuming that the crash does not result in file system corruption. **vxconfigd** can be compiled such that logging defaults either to enabled or disabled. If it defaults to disabled, then it can be turned on with **–x log**; otherwise, it can be turned off with **–x nolog**.

For Solaris, logging is disabled by default. If enabled, the default log file location is **/var/vxvm/vxconfigd.log**.

logfile=*logfilename*
Specify an alternate location for the **vxconfigd** logfile. This option implies **–x log**.

cleartempdir
This option causes the **/var/vxvm/tempdb** directory to be removed and recreated. This directory stores configuration information that is cleared on reboots (or cleared for specific disk groups on import and deport operations). If the contents of this directory become corrupt,

such as due to a disk I/O failure, then **vxconfigd** will fail to start up if it is killed and restarted. Such a situation can be cleared by starting **vxconfigd** with **–xcleartempdir**. This option has no effect if **vxconfigd** is not started in *enabled* mode.

NOTE: It is advisable to kill any running operational utilities (**vxvol**, **vxsd**, or **vxmend**) before using the **–x cleartempdir** option. Failure to do so may cause those commands to fail, or may cause disasterous but unchecked interactions between those commands and the issuance of new commands. It is okay to use this option while running the VERITAS Visual Administrator (**vxva**), or while VxVM background daemons are running (**vxsparecheck** or **vxnotify**).

stub This **vxconfigd** invocation will not communicate configuration changes to the kernel. It is typically used as a demonstration mode of operation for **vxconfigd**. In most aspects, a stubbed **vxconfigd** will act like a regular **vxconfigd**, except that disk devices can be regular files and volume and plex device nodes are not created. A stubbed **vxconfigd** can run concurrently with a regular **vxconfigd**, or concurrently with any other stubbed **vxconfigd** processes, as long as different rendezvous, **volboot**, and disk files are used for each concurrent process.

Other Volume Manager utilities can detect when they are connected to a **vxconfigd** that is running in stubbed mode. When a utility detects a stubbed-mode **vxconfigd**, it will normally stub out any direct use of volume or plex devices, itself. This allows regular utilities to be used for making configuration changes in a testing environment that runs without any communication with the kernel or creation of real volume or plex devices.

boot=*volboot_path*

Specify the pathname to use for the **volboot** file. This is primarily of use with the **stub** debug option. The **volboot** file contains an initial list of disks that are used to locate the root disk group. It also contains a host ID that is stored on disks in imported disk groups to define ownership of disks as a sanity check for disks that might be accessible from more than one host.

devprefix=*prefixdir*

Specify a directory pathname to prefix for any disk device accessed by **vxconfigd**. For example, with **devprefix=/tmp**, any access to a raw disk device named **c2t1d0s2** would actually be directed to the file **/tmp/dev/rdsk/c2t1d0s2**. In stubbed-mode, **vxconfigd** can operate with such files being regular files. **vxconfigd** only requires entries in the *prefixdir* **/dev/rdsk** directory in stubbed mode.

tracefile=*file*

Log all possible tracing information in the given file.

synctrace
Flush tracefile data to disk, with **fsync(2)**, to ensure that the last entry will be included in the file, even if the system crashes.

noautoconfig
Normally, **vxconfigd** will automatically configure disk devices that can be found by inspecting kernel disk drivers. These *auto_configured* disk devices are not stored in persistent configurations, but are regenerated from kernel tables after every reboot. Invoking **vxconfigd** with **–x noautoconfig** prevents the automatic configuration of disk devices, forcing the Volume Manager to use only those disk devices configured explicitly using **vxdisk define** or **vxdisk init**.

vfstab=*vfstabfile*
Specify the pathname to use for the **vfstab** file. Normally, **/etc/vfstab** is used. The **vfstab** file is used to determine whether the **/usr** file system is mounted on a volume, and to determine the name of that volume.

–D *diag_portal*
Specify a rendezvous file pathname for diagnostic operation connections to **vxconfigd**. By default, **/etc/vx/vold_diag** is used. The diagnostic portal exists in both the *enabled* and *disabled* operating modes.

–R *request_portal*
Specify a rendezvous file pathname for regular configuration and query requests. By default, this is **/etc/vx/vold_request**. The regular request portal exists only when **vxconfigd** is operating in *enabled* mode.

DIAGNOSTICS

If errors are encountered, **vxconfigd** writes diagnostic messages to the standard error output. Some serious errors will cause **vxconfigd** to exit. If an error is encountered when importing the **rootdg** disk group during a normal startup, **vxconfigd** will enter *disabled* mode. The error messages section of the *VERITAS Volume Manager System Administrator Guide* should be consulted for a description of the diagnostics and the suggested course of action.

Defined exit codes for **vxconfigd** are:

(0) The requested startup mode completed successfully. This is returned if **–f** is not used to startup **vxconfigd** as a foreground process. If **vxconfigd** is started as a foreground process, then it will exit with a zero status if **vxdctl stop** is used to cause **vxconfigd** to exit.

(1). The command line usage is incorrect.

(2). *Enabled*-mode operation was requested, but an error caused **vxconfigd** to enter *disabled* mode instead. This is also returned for *boot*-mode operation if startup failed. However, with *boot*-mode operation, the background **vxconfigd** process exits as well.

(3). The **–k** option was specified, but the existing **vxconfigd** could not be killed.

(4). A system error was encountered that **vxconfigd** cannot recover from. The specific operation that failed is printed on the standard error output.

(5). The background **vxconfigd** process was killed by a signal before startup completed. The specific signal is printed on the standard error output.

(6). A serious inconsistency was found in the kernel, preventing sane operation. This can also happen because of version mismatch between the kernel and **vxconfigd**.

(7). The **–r resest** option was specified, but the volume manager kernel cannot be reset. Usually this means that a volume is open or mounted.

(8). An interprocess communications failure (usually a STREAMS failure). Has made it impossible for **vxconfigd** to take requests from other utilities.

(9). Volumes that must be started early by **vxconfigd** could not be started. The reasons, and possible recovery solutions, are printed to the standard error output. For Solaris, the early-started volumes are the root and **/usr** file systems, if either or both of those file systems are defined on volumes.

FILES

/dev/vx/dsk	directory containing block device nodes for volumes.
/dev/vx/rdsk	directory containing raw device nodes for volumes.
/etc/vx/volboot	file containing miscellaneous boot information. See **vxdctl**(1M) for more information on this file.
/var/vxvm/tempdb	directory containing miscellaneous temporary files. Files in this directory are recreated after reboot.

SEE ALSO

vxintro(1M), **vxdctl**(1M), **syslog**(3), **syslogd**(1M), **syslog.conf**(4B).

NAME

vxdctl – control the volume configuration daemon

SYNOPSIS

vxdctl init [*hostid*]
vxdctl hostid *hostid*
vxdctl add disk *accessname* **[** *attr*[**=***value*]] ...
vxdctl rm disk *accessname* ...
vxdctl list
vxdctl enable
vxdctl disable
vxdctl [**–k**] **stop**
vxdctl mode

DESCRIPTION

The **vxdctl** utility manages some aspects of the state of the volume configuration daemon, **vxconfigd**, and manages configuration aspects related to bootstrapping the **rootdg** disk group configuration.

A key part of the state of **vxconfigd** and of bootstrapping the **rootdg** disk group is the **volboot** file. This file contains a host ID that is used by the Volume Manager to establish ownership of physical disks. This host ID is used to ensure that two or more hosts that can access disks on a shared SCSI bus will not interfere with each other in their use of those disks. This host ID is also important in the generation of some unique ID strings that are used internally by the Volume Manager for stamping disks and disk groups.

The **volboot** file also contains a list of disks to scan in search of the **rootdg** disk group. At least one disk in this list must be both readable and a part of the **rootdg** disk group, or the Volume Manager will not be able to start up correctly.

vxconfigd operates in one of three modes: *enabled*, *disabled*, or *booted*. The *enabled* state is the normal operating state. Most configuration operations are allowed in the *enabled* state. Entering the *enabled* state imports all disk groups, and begins the management of device nodes stored in the **/dev/vx/dsk** and **/dev/vx/rdsk** directories.

In the *disabled* state, **vxconfigd** does not retain configuration information for the imported disk groups, and does not maintain the volume and plex device directories. Most operations are disallowed in the *disabled* state. Certain failures, most commonly the loss of all disks or configuration copies in the **rootdg** disk group, will cause **vxconfigd** to enter the *disabled* state automatically.

The *booted* state is entered as part of normal system startup, prior to checking the root file system (see **fsck**(1M). Entering the *booted* mode imports the **rootdg** disk group, and then waits for a request to enter the *enabled* mode. The volume and plex device node directories are not maintained in *booted* mode, because it may not be possible to write to the root file system.

The action performed by **vxdctl** depends upon the keyword specified as the first operand. Supported operations are:

vxdctl init

Reinitialize the **volboot** file with a new host ID, and with an empty list of disks. If a *hostid* operand is specified, then this string is used; otherwise, a default host

ID is used. The default host ID is normally taken as the network node name for the host (see **uname**(1M)). On systems with a hardware-defined system ID, the default host ID might be derived from this hardware ID.

vxdctl hostid

Change the host ID in the **volboot** file and on all disks in disk groups currently imported on this machine. It may be desirable to change the Volume Manager host ID for your machine if you are also changing the network node name of your machine.

If some disks are inaccessible at the time of a **hostid** operation, it may be necessary to use the **vxdisk clearimport** operation to clear out the old host ID on those disks when they become reaccessible. Otherwise, you may not be able to re-add those disks to their disk groups.

NOTE: Some care should be taken when using this command. If the system crashes before the **hostid** operation completes, some disk groups may not reimport automatically.

vxdctl add disk

Add to the list of disks in the **volboot** file. Disks are specified based on their disk access name. This name identifies the physical address of the disk. For example, to add disk 0 at SCSI target 1 on SCSI controller 0, you might use the command:

vxdctl add disk c0t1d0

This will add the disk **c0t1d0s2**. The **s2** part is assumed. If disk with another slice is to be added the slice will have to be specified like in:

vxdctl add disk c0t1d0s3

If there is a disk access record in the **rootdg** configuration for the named disk, then configuration parameters are taken from that record. Otherwise, it may be necessary to specify some attributes to **vxdctl add disk**.

vxdctl rm disk

Remove one or more disks from the **volboot** file. Disks are specified based on the name used in the corresponding **vxdctl add disk** operation.

vxdctl list

List the contents of the **volboot** file. This list includes the host ID, some sequence numbers, and the list of disks and disk attributes stored in the **volboot** file.

vxdctl enable

Request that **vxconfigd** enter *enabled* mode, import all disk groups, and rebuild the volume and plex device node directories. This operation can be used even if **vxconfigd** is already in *enabled* mode. The primary purpose for using this operation when in *enabled* mode is to rebuild the volume and plex device nodes. This operation will also cause **vxconfigd** to scan for any disks that were newly added since **vxconfigd** was last started. In this manner, disks can be dynamically configured to the system and then recognized by the Volume Manager.

vxdctl disable

Request that **vxconfigd** enter *disabled* mode. This may be necessary to perform

some maintenance operations. This does *not* disable any configuration state loaded into the kernel. It only prevents further configuration changes to loaded disk groups until **vxconfigd** is re-enabled.

vxdctl stop

Request that **vxconfigd** exit. This may be necessary to reset the Volume Manager, such as using the **-r reset** option to **vxconfigd**. This does *not* disable any configuration state loaded into the kernel. It only affects the ability to make configuration changes until **vxconfigd** is restarted.

vxdctl mode

Print the current operating mode of **vxconfigd**. The output format is:

mode: *operating_mode*

where *operating_mode* is either **enabled**, **disabled**, **booted**, or **not-running**.

SEE ALSO

vxintro(1M), **vxconfigd**(1M), **vxdg**(1M), **vxdisk**(1M).

NAME

vxdg – manage Volume Manager disk groups

SYNOPSIS

vxdg init *groupname* [*nconfig=config-copies*] [*nlog=log-copies*] [*minor=base-minor*] [*medianame=*]*accessname* ...
vxdg [**–g** *diskgroup*] [**–f**] **reminor** [*diskgroup*] *new-base-minor*
vxdg [**–tfC**] [**–n** *newname*] **import** *diskgroup*
vxdg [**–n** *newname*] [**–h** *newhostid*] **deport** *diskgroup* ...
vxdg [**–g** *diskgroup*] [**–k**] **adddisk** [*medianame=*]*accessname* ...
vxdg [**–g** *diskgroup*] [**–k**] **rmdisk** *medianame* ...
vxdg [**–q**] **list** [*diskgroup* ...]
vxdg [**–g** *diskgroup*] [**–qa**] **free** [*medianame* ...]
vxdg flush [*diskgroup* ...]
vxdg [**–g** *diskgroup*] [**–k**] **repldisk** *unassoc-medianame=spare-medianame* ...

DESCRIPTION

The **vxdg** utility performs basic administrative operations on disk groups. Operations include the creation of disk groups, the addition of disks to a disk group, and disk group imports and deports. The behavior of the **vxdg** utility depends upon the keyword specified as the first operand.

A *diskgroup* argument can be either a disk group name or a disk group ID. A *groupname* argument is a disk group name, not a disk group ID. An *accessname* argument refers to a system-dependent disk access name (also referred to as a disk device name), as stored in the root configuration by the **vxdisk** utility. A *medianame* argument is an administrative name defined within a disk group.

An *accessname* argument names a disk access record (essentially a device address specification) used to access the disk. If the slice number extension in the disk access record name is not included in the accessname, **s2** is assumed by default. If any other slice is required then it should be included in the accessname as in **c0t1d0s4**.

A *medianame* argument names the disk media record used to define the disk within the disk group.

Supported operations are:

vxdg init

Define a new disk group composed of the indicated disks, identified by disk access names. This involves assigning an internal unique ID to the group, storing a pointer to that group in the root configuration, storing a reference to the group on all of the named disks that have a disk header, and storing a disk group record in the disk group's configuration database. At least one of the disks specified must have space allocated for a configuration copy.

If a *medianame* is specified for use with a particular disk, then that *medianame* will name the disk media record used to reference the disk within the disk group, for the **rmdisk** operation and for subdisk creations. If no *medianame* is specified, then the disk media name defaults to *accessname*. See **vxdisk**(1M) for discussion of definition and initialization of disk access records for use with this operation.

The **init** operation can be used to initialize a root disk group configuration, which

is identified by the special name **rootdg**. If any database locations are listed in the **volboot** file, then as a special case for initializing **rootdg**, no disk specifications are allowed. Disks should be initialized and added to the disk group as the first operations after creating **rootdg**. Some or all disks added to the **rootdg** disk group should also be added to the **volboot** bootstrap file (see **vxdctl**(1M)).

The **nconfig** and **nlog** operands can be used to configure the number of configuration database copies and kernel log copies that are maintained for a disk group. The *config-copies* and *log-copies* values are either a decimal number, including 0 or –1, or they can be **all** or **default**. A value of **all** or –1 signifies that all configuration or log copies on all disks in the disk group will be maintained. A value of **default** or 0 (this is also the default value) signifies that the Volume Manager will manage copies that are distributed in a reasonable pattern throughout the disks and controllers on the system. Any other number requests that a particular number of copies be maintained (or all copies, if that number is larger than the number of available configuration or log copies on all disks).

When a specific number or (**default**) is requested, configuration copies are scattered approximately evenly through the disk controllers on the system. If SCSI disks with multiple disks per target are found, then each such target is treated similarly to a controller (i.e., configuration copies are evenly distributed between such targets). With the **default** policy, one configuration or log copy is maintained for each controller, and one configuration or log copy is also maintained for each SCSI target that has multiple disks; if this does not result in allocating at least 4 copies, then additional copies are spread through the controllers and targets.

Refer to **vxdisk**(1M) for more information on configuration and log copies, and for information on how to create them.

NOTE: If a policy other than **all** is used, then some disks will not have up-to-date, online configuration and log copies. As a result, it is possible that some number of disk failures will leave a disk group unusable, even if some disks in the disk group remain usable. The **default** policy allocates a sufficient number of copies, in a sufficient spread of locations, that such a scenario is very unlikely to occur.

Since disk groups can be moved between systems, it is desirable that device numbers used for volumes be allocated in separate ranges for each disk group. That way, an administrator can choose ranges such that all disk groups in a group of machines can be moved around without causing device number collisions. This problem arrises because the volume manager stores device numbers in disk group configurations, so that the same numbers can be used after a reboot (which is necessary for use with NFS, which requires persistency of device numbers). If two systems use the same device numbers for a set of volumes, and if a disk group from one machine is moved to the other, then the volume manager may be forced to temporarily remap some devices.

A base volume device minor number can be set for a disk group with the **minor** operand. Volume device numbers for a disk group will be chosen to have minor numbers starting at this base minor number. Minor numbers (in Solaris) can range up to 131071, so if it is presumed that no more than 1000 volumes would ever be created in any one disk group, then 131 different ranges of minor numbers are available for different disk groups. A reasonably sized range should be left at the end for temporary device number remappings (in the event that two device numbers do, still, conflict).

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

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

vxdg reminor

Change the base minor number for a disk group, and renumber all devices in the disk group to a range starting at that number. If the device for a volume is open, then the old device number will remain in effect until the system is rebooted or until the disk group is deported and re-imported. Also, if you close an open volume, then the can execute **vxdg reminor** again to cause the renumbering to take effect without rebooting or reimporting.

A new device number may also overlap with a temporary renumbering for a volume device, which will also require a reboot or reimport for the new device numbering to take effect. A temporary renumbering can happen in the following situations: when two volumes (for example, volumes in two different disk groups) share the same permanently assigned device number, in which case one of the volumes is renumbered temporarily to use an alternate device number; or when the persistent device number for a volume was changed, but the active device number could not be changed to match. The active number may be left unchanged after a persistent device number change either because the volume device was open, or because the new number was in use as the active device number for another volume.

vxdg will fail if you try to use a range of numbers that is currently in use as a persistent (not a temporary) device number. You can force use of the number range with use of the **–f** option. With **–f**, some device renumberings may not take effect until a reboot or a re-import (just as with open volumes). Also, if you force volumes in two disk groups to use the same device number, then one of the volumes will be temporarily renumbered on the next reboot. Which volume device will be renumbered should be considered random, except that device

numberings in the **rootdg** disk group take precedence over all others.

The **–f** option should be used only when swapping the device number ranges used by two or more disk groups. To swap the number ranges for two disk groups, you would use **–f** when renumbering the first disk group to use the range of the second disk group. Renumbering the second disk group to the first range will not require use of **–f**.

vxdg import

Import a disk group to make the specified disk group available on the local machine. This will make any configuration information stored with the disk group accessible, including any disk and volume configurations. The disk group to import is indicated by the *diskgroup* argument, which can be either an administrative disk group name or a disk group unique ID.

Normally, a disk group will not be imported if some disks in the disk group cannot be found by the local host. The **–f** option can be used to force an import if, for example, one of the disks is currently unusable or inaccessible.

NOTE: Care must be taken when using the **–f** flag, since it can cause the same disk group to be imported twice from disjoint sets of disks, causing the disk group to become inconsistent.

When a disk group is imported, all disks in the disk group are stamped with the host's host ID. Normally, a disk group cannot be imported if any of its disks are stamped with a non-matching host ID. This provides a sanity check in cases where disks can be accessed from more than one host.

If it is certain that a disk is not in use by another host (such as because a disk group was not cleanly deported), then the **–C** option can be used to clear the existing host ID on all disks in the disk group as part of the import. A host ID can also be cleared using **vxdisk clearimport**.

A new name can be given to the disk group on import using **–n** *newname*. If **–n** is used with the **–t** option, then the stored name of the disk group will remain unchanged, but the disk group will be known to the importing host under the new name; otherwise, the name change will be permanent.

Normally, an imported disk group will be reimported automatically when the system is rebooted, if at least some of the disks in the disk group remain accessible and usable. This can be disabled using the **–t** option, which causes the import to persistent only until the system is rebooted.

As an example of the use of **–n** and **–t**, a **rootdg** disk group from one host can be imported on a second host, operations can be performed on the second host (such as making repairs to the root volume), and the disk group can be given back to the originating host, which can then be rebooted on the repaired disk group. To do this, identify the disk group ID for the **rootdg** disk group with **vxdisk –s list**, and use that disk group to import that **rootdg** using **–C** to clear import locks, **–t** for a temporary name, and **–n** to specify an alternate name (to avoid collision with the **rootdg** disk group on the second host). After repair, deport the disk group using **–h** (described below) to restore the import lock from

the first host.

vxdg deport

Disable access to the specified disk group. A disk group cannot be deported if any volumes in the disk group are currently open. When a disk group is deported, the host ID stored on all disks in the disk group will be cleared (unless a new host ID is specified with **–h**), so the disk group will not be reimported automatically when the system is rebooted.

A disk group can be renamed on deport by specifying a new disk group name with **–n** newname. A lock can be assigned to an alternate host by specifying the host ID [see **vxdctl**(1M)] of the alternate host. This allows the disk group to be auto-imported when the alternate host reboots. For example, the **–n** and **–h** options can be combined to export a disk group to be used as the **rootdg** disk group for a new machine.

vxdg adddisk

Add additional disks to a disk group (**rootdg** by default). The disk must not already be part of an imported disk group. The *accessname* component to a disk specification argument names a disk access record (essentially a device address specification) used to access the disk. If a *medianame* component is specified, then it names the disk media record used to define the disk within the disk group. If no *medianame* component is specified, then the disk media record will have the same name as the disk access record.

Adding a disk to a disk group causes the disk group's configuration to be copied onto the disk (if the disk has regions for configuration copies). Also, the disk is stamped with the system's host ID, as defined in the **volboot** file.

If the **–k** flag is specified, then the disk media name must represent a disk media record that was previously dissociated from its disk access record with **–k rmdisk**; otherwise, a new disk media record will be created to represent the disk. With the **–k** option, plexes requiring recovery will be flagged as stale.

vxdg rmdisk

Remove the specified disks from a disk group (**rootdg** by default). The last disk cannot be removed from a disk group. It is not possible to remove the last disk containing a valid disk group configuration or log copy from its disk group.

Normally, the operation will fail if subdisk records point to the named disk media records. However, if the **–k** flag is specified, then the disk media records will be kept, although in a *removed* state, and the subdisk records will still point to them. The subdisks, and any plexes that refer to them, will be unusable until the disk is re-added using the **–k** option to the **adddisk** operation. Any volumes that become unusable, because all plexes become unusable, will be disabled.

vxdg list

List the contents of disk groups. If no *diskgroup* arguments are specified, then all disk groups are listed in an abbreviated one-line format. If *diskgroup* arguments are specified, then a longer format is used to indicate the status of the disk group, and of the disk group configuration.

If the **–q** option is specified, then no header is printed describing output fields. This option has no effect with the long formats generated with *diskgroup* arguments.

vxdg free

List free space that can be used for allocating subdisks. If a disk group is specified, limit the output to the indicated disk group, otherwise list space from all disk groups. If disks are specified, by disk media name, then restrict the output to the indicated disks. A region of free space is identified by disk media name, a physical device tag, an offset relative to the beginning of the public region for the media, and a length.

The physical device tag is a reference that indicates which physical device the disk media is defined on. It appears as a truncated disk access name. If a particular physical device is split into several Volume Manager disk objects, then the device tag for each Volume Manager disk object will be the same. Device tags can be compared to identify space that is on the same or on different physical disks.

If the **–q** option is specified, then no header is printed describing output fields. If the **–a** option is specified, then space on spare disks (which is not really allocatable) is listed in addition to regular free space; otherwise, space on spare disks is *not* listed.

vxdg flush

Rewrite all disk on-disk structures managed by the Volume Manager for the named disk groups. This rewrites all disk headers, configuration copies, and kernel log copies. Also, if any configuration copies were disabled, for example as a result of I/O failures, this will rewrite those configuration copies and attempt to enable them.

vxdg repldisk

Dissociate the DA record from the DM record named by *spare-medianame* and reassociate it with the unassociated DM record named by *unassoc-medianame*. Both *unassoc-medianame* and *spare-medianame* must be members of the disk group named by the *diskgroup* argument (**rootdg** by default). However, if the **–k** flag is specified, then the disk media records for the *spare-medianame* will be kept, although in a *removed* state.

SEE ALSO

vxintro(1M), **vxconfigd**(1M), **vxdisk**(1M), **vxplex**(1M), **vxvol**(1M).

NAME

vxdisk – define and manage Volume Manager disks

SYNOPSIS

vxdisk [**–f**] **init** *accessname* [*attribute* ...]
vxdisk [**–f**] **define** *accessname* [*attribute* ...]
vxdisk offline *accessname* ...
vxdisk online *accessname* ...
vxdisk rm *accessname* ...
vxdisk [**–g** *diskgroup*] [**–qs**] **list** [*disk* ...]
vxdisk clearimport *accessname* ...
vxdisk [**–g** *diskgroup*] **check** *disk* ...
vxdisk [**–g** *diskgroup*] **addregion** *region_type disk offset length*
vxdisk [**–g** *diskgroup*] **rmregion** *region_type disk offset* [*length*]
vxdisk [**–g** *diskgroup*] **set** *disk* [*attribute* ...]

DESCRIPTION

The **vxdisk** utility performs basic administrative operations on disks. Operations include initializing and replacing disks, as well as taking care of some book-keeping necessary for the disk model presented by the Volume Manager.

vxdisk accesses disks based on disk access names, which are system-specific names that relate to disk addresses. For System V Release 4, disk access names are normally of the form **c***C***t***T***d***D***s***S*, which define a controller number (*C*), a SCSI target ID (*T*), a SCSI logical unit number, and a partition number (*S*). While typing on the command line if the partition number is not specified, **s2** is assumed which represents the whole disk. If any other partition is required, it should be specified as in **c0t1d0s4**. Special devices, such as internal RAM disks, may use different forms for disk access names. All disk access names used with System V Release 4 relate directly to device node names in the **/dev/dsk** and **/dev/rdsk** directories.

Operations that take an *accessname* argument (see the SYNOPSIS section) accept only disk access names, as defined in the previous paragraph. Operations that take a *disk* argument can take disk access names or disk media names (e.g., **disk01**). For such operations, a disk group can be specified with **–g** to disambiguate disk media names that are used in more than one disk group.

Physical disks in the Volume Manager are presumed to be movable, and are usually identified by a unique disk ID stored on the physical disk, rather than by disk device node. This allows disks to be moved to different SCSI target IDs or to different controllers without affecting correct operation.

The Volume Manager maintains known disk device address information in a set of disk access records, which are stored in the **rootdg** disk group configuration. These records are named, based on the disk access name. These disk access records are normally used solely to identify which physical disks exist, based on disk IDs stored on the disks themselves. Operations for **vxdisk** other than **init** and **define** require specification of defined disk access records.

Physical disks contain public regions, which are used for allocating subdisks. They can also contain private regions, which are used for storing private Volume Manager information. Private regions are structured regions, and are maintained entirely by the Volume Manager. Private regions contain the following structures:

Disk header

Each private region contains exactly two copies of a disk header, which defines the unique disk ID, disk geometry information, and disk group association information. Two copies are created so that one copy can be lost (due to I/O failures), without losing the ability to use the disk. The primary copy of the disk header is stored in block zero of the private region. The alternate copy is stored within the first 256 sectors. If the primary copy is unreadable or unusable, the Volume Manager will search the first 256 sectors of the private region for the alternate copy.

Table of contents

A linked list of blocks, pointed to by the disk header, that define additional structures in the private and public regions. The table of contents blocks define disk group configuration copy locations, log copy locations, and reserved regions carved from the public region. Each link block in the table of contents is replicated at the beginning and end of the private region. If the primary copy of any one link block is unreadable or unusable, the alternate copy of that link is used.

Configuration copies

A disk normally contains one disk group configuration copy, according to the number specified when the disk was initialized using the **vxdisk init** operation (explained later). When a disk is added to a disk group, the disk group's persistent configuration records are written to each copy. For disks that are not associated with a disk group, the space allocated for configuration copies is unused. Each disk group requires at least one usable configuration copy. Preferably there should be at least four copies, allocated between at least two disks. This allows one disk to be lost totally, while still preserving sufficient redundancy for recovering from simple read failures.

Disk group log copies

A disk normally contains one disk group log copy. The number of log copies is set to the same as the number of configuration copies for the disk as explained in the *Configuration copies* section above. These logs are written by the kernel when certain types of actions are performed: transaction commits, plex detaches resulting from I/O failures, total dirty-region-log (DRL) failures, the first write to a volume, and volume close. After a crash or a clean reboot, this log information is used to recover the state of a disk group just prior to the crash or reboot. Each disk group requires at least one usable disk group log copy. As with configuration copies, it is preferable to have at least four log copies, allocated between at least two disks.

For a single disk, the disk header and the table of contents blocks are critical data structures. At least one copy of the disk header, and at least one copy of each table of contents block, must be readable and usable, or else the disk itself is unusable and will have to be

reinitialized.

Within disk groups, disk group configuration and log copies are critical data structures. At least one complete configuration copy and log copy must be readable and usable, or the disk group is unusable and will have to be reinitialized from scratch.

All disk group association information is stored in the disk header within private regions. This information consists of a disk group name, disk group unique ID, and a host ID. When the system boots, the Volume Manager scans for disks that are stamped with the system's host ID. Each represented disk group is imported automatically. Disks with a non-matching host ID are not imported automatically, and cannot be used until the host ID is cleared with the **clearimport** operation.

The behavior of the **vxdisk** utility depends upon the keyword specified as the first operand. Supported operations are:

vxdisk init

Initialize regions of a disk used by the Volume Manager. This involves installing a disk header and writing an empty configuration on the disk. The *accessname* operand identifies the disk. Normally, this command will fail if the disk already contains an apparently valid disk header. The **–f** option can be used to override this and to force initialization of the disk. A disk that is a member of an imported disk group cannot be initialized.

The **vxdisk init** operation creates a disk access record for a disk (if one does not already exist), and sets its state to **online**. Disks can be initialized when the root configuration is disabled, in which case the disk header will be initialized, but the disk will not be added to the permanent list of known disks until the root configuration is enabled.

Any *attribute* operands override default values assigned for various disk attributes. Some attributes that can be set are:

type=*disk_type*

The disk device access type, which is a system-specific name identifying a class of strategies for accessing disks and for managing private and public regions. For example, disk types could indicate network disks, or could indicate a volatile RAM disk that may not require the storage of any private data, or could be for a hard disk without separate partitions. If the disk accessname is of the form **c***C***t***T***d***D***s2* or **c***C***t***T***d***D**, (where the capital letters represent a supplied digit) then the disk type defaults to **sliced**; otherwise, the disk type defaults to **simple**.

The various disk types support additional attributes for the **init** operation. See the definition for each disk type, below.

offline The device will be left in the **offline** state, initially. This is used only if this operation is defining a new disk access record.

vxdisk define

Define a disk access record, but do not initialize it. In order for the Volume Manager to scan a disk, a disk access record must be defined for it. Thus, if you

want to see what is on a new disk or you want to move a disk with a valid disk group from one system to another, you will need to use **vxdisk define** to make it accessible first. You can use **vxdisk list** to see what is on the disk, or **vxdg import** to import a disk group that is on the disk.

Attributes can be specified to define the access characteristics of the disk device. Some attributes that can be set are:

type=*disk_type*
The disk device access type. See the **init** operation definition for more details.

The various disk types support additional attributes for the **define** operation. See the definition for each disk type, below.

offline If specified, the disk will be created in the **offline** state.

Normally, a **define** operation will fail if the specified disk device is invalid, such as because no such disk currently exists. The **–f** option can be used to force definition of an unusable disk. This can be useful if, for example, the disk device could be useful after a reboot. For example, if you intend to add a new controller and intend to move some existing disks to the new controller, you may need to define the new disk device addresses, even though they will not be usable until you shutdown and reconfigure your disks.

vxdisk offline
Declare the disk devices named by the *accessname* arguments to be in the **offline** state. This disables checking of the disk in searching for particular disk IDs, or for the set of disks in a particular disk group. This operation cannot be applied to disks that are members of an imported disk group.

A disk should be offlined if the disk is not currently accessible, and if accessing the disk has a negative impact on the system. For example, disk drivers on a few operating systems can cause system panics or hangs if an attempt is made to access disks that are not accessible. In other operating systems, attempts to access inaccessible drives may take several seconds or minutes before returning a failure.

vxdisk online
Clear the **offline** state for a disk device. This re-enables checking of the disk when searching for disk IDs, or for members of a disk group. This can be used for disks that are already in the **online** state, provided that they are not in imported disk groups. All internal information for an already **online** state disk is regenerated from the disk's private region.

vxdisk rm
Remove disk access records, by name.

vxdisk list
List detailed disk information on the specified disks. If no *disk* arguments are specified, then print a one-line summary for all disk access records known to the system. If *disk* arguments are specified, then print a full description of the

contents of the disk header and of the table of contents for each named disk. If no *disk* arguments are specified, but a disk group is specified with **–g**, then list only those disks added to the specified disk group.

If the **–s** option is specified, then list important information from the disk header. With **–s**, the output format is the same whether or not *accessname* arguments are specified. The information printed with **–s** includes the disk ID, the host ID (if the disk is or was imported), and the disk group ID and disk group name (if the disk is a member of a disk group).

If the **–q** option is specified, then no header is printed describing output fields. This option has no effect with the long formats generated with **–s** or with *accessname* arguments.

vxdisk clearimport

Clear the host-specific import information stored on the indicated disks, and in the configurations stored on those disks. This command may be necessary in cases where import information stored for a disk group becomes unusable, due to host failures, or due to a disk group being moved from one machine to another.

This operation cannot be applied to disks that are in imported disk groups.

vxdisk check

Validate the usability of the given disks. A disk is considered usable if the volume manager can write and read back at least one of the disk headers that are stored on the disk. If a disk in a disk group is found not to be usable, then it is detached from its disk group and all subdisks stored on the disk become invalid until the physical disk is replaced or the disk media record is reassigned to a different physical disk.

vxdisk addregion

Add a new entry to the table of contents in a disk's private region. The new entry defines a region of disk that is relative to the public partition, and that is reserved for a particular use. The *offset* and *length* operations indicate the location and extent of the region. Currently, the only region type that can be defined is:

reserve Mask out a region of disk that should be reserved for non-volume-manager purposes. This could be used, for example, to mask out a boot file system that cannot be used for subdisk allocation, or to mask out a region containing blocks that are used for bad-block or bad-track replacement.

Adding a region will fail if a subdisk or region is already allocated over the requested region.

The **addregion** functionality is currently unimplemented for any of the existing disk types.

vxdisk rmregion

Free a region of space that is allocated in the private or public partition for a particular use. Space that is freed from the public partition becomes usable for

subdisk creation. The arguments to **rmregion** must match the arguments used when adding the region with **vxdisk addregion** except for the optional *length* argument which can be excluded for the remove.

The **rmregion** functionality is currently unimplemented for any of the existing disk types.

vxdisk set

Change some set of attributes for a disk. The attributes are either simple names (used to turn on an on / off attribute), or can be of the form *attrname=value*, to indicate a value for a particular attribute.

The **set** functionality is currently unimplemented for any of the existing disk types.

DISK TYPES

Three disk types are provided with the base VERITAS Volume Manager. Additional types may be added for use with particular operating systems. The default is a **sliced** type for disk access names ending in **s2**. If the slice component is not specified, then too type defaults to sliced. If any other slice number is supplied, then the default type becomes **simple**.

Nopriv disks

The simplest disk type is **nopriv**, which defines a disk that has no private region, and that consists only of space for allocating subdisks. Configuration and log copies cannot be stored on such disks, and such disks do not support reserved regions defined with **vxdisk addregion**. Also, such disks are not self identifying. Because **nopriv** disks are not self identifying, the Volume Manager cannot track the movement of such disks on a SCSI chain or between controllers.

nopriv devices are most useful for defining special devices (such as volatile RAM disks) that you wish to use with the Volume Manager, but that can't store private regions. A RAM disk cannot store a meaningful private region, because data written to a RAM disk may not survive a reboot.

Initializing a **nopriv** device with **vxdisk init** creates a disk access record in the **rootdg** configuration, but does not write to the disk. The disk ID for **nopriv** devices is stored in the disk access record in the **rootdg** configuration.

Attributes that can be used with the **vxdisk init** and **define** operations for the **nopriv** device type are:

publen=*length* or **len=***length*

The usable length of the device. This is required if there is no system-defined procedure for determining the disk length; otherwise, a suitable default will be computed.

puboffset=*offset* or **offset=***offset*

The offset within the device for the start of the usable region. This defaults to 1. This can be used if it is necessary to skip over some region reserved by the operating system. If an offset is specified, then the default disk length is adjusted accordingly.

volatile

If this attribute is specified, the disk is considered to have volatile contents (i.e., the disk contents are not expected to remain consistent across a system reboot). Subdisks and plexes defined on disks with the **volatile** attribute will inherit that attribute. The **vxvol start** operation interprets volatile plexes as requiring a complete revive from other plexes in the same volume.

The **vxdisk define** operation, with the **nopriv** device type, takes the same attributes as the **init** operation. In addition, **define** takes the following attribute:

diskid=*newdiskid*
This attribute will set the disk ID to the *newdiskid* value in the disk access record for the **nopriv** disk.

Sliced and simple disks

Two disk types are provided that support disk private regions: **simple** and **sliced**. The **simple** type presumes that the public and private regions are stored on the same disk partition, with the public region following the private region. The **sliced** type presumes that the public and private regions are stored on different disk partitions.

For the **sliced** type, if the access name ends in **s2**, then the the public and private regions default to disk partitions with the tags 14 and 15, respectively. If the disk access name ends in **s** followed by some octal digit other than 0 and 1, the public region defaults to the named partition, and the private region defaults to the immediately following partition. For example, with an access name of **c0t1d0s4**, the public region defaults to partition 4 (octal **4**) and the private region defaults to partition 5. For an accessname of **c0t1d0s1**, the public region defaults to partition 1 and the private region defaults to partition 3.

Attributes that can be defined with **vxdisk define** for **simple** or **sliced** types are:

pubslice=*number* or **slice=***number*
Define the partition number to use for the public partition. This can be used only with the **sliced** type.

privslice=*number*
Define the partition number to use for the private partition. This can be used only with the **sliced** type.

privoffset=*offset*
Specify the offset from the beginning of the partition containing the private region to the beginning of the private region. This defaults to zero for both **simple** and **sliced** types. It is strongly advised that this option never be used unless absolutely necessary. Most information for disks can be determined from the disk header stored at the beginning of the private region. The private region offset cannot be determined from the disk. As a result, specifying a private region offset adds an undesirable dependence between a disk access record and a specific physical disk.

In addition to the above attributes, **vxdisk init** takes the following attributes:

publen=*length* or **len=***length*
Specify the length of the public region. If this is not specified, then the length of the private region is computed from available partition table information. If no such information is available, a public region length must be specified in this

command. The default public region length is adjusted to account for the private region, or for any specified public or private region offsets.

puboffset=*offset* or **offset=***offset*
Specify the offset from the beginning of the partition containing the public region to the beginning of the public region. For the **sliced** type, this defaults to one. For the **simple** type, this defaults to the first block past the end of the private region.

privlen=*length*
Specify the length of the private region. If this is not specified, then a default is chosen. For the **sliced** type, the default is computed from available partition table information. For the **simple** type, the default size is 528 blocks. With the **sliced** type, if no partition information is available, a private region length must be specified in this command.

nconfig=*count*
The number of configuration copies to store on the disk. This defaults to 1. This can be set to 0 to indicate that no configurations will be stored on the disk. The Volume Manager will automatically enable and disable the config copy. It will maintain a level of redunancy in configuration copies that will allow the configuration to be recovered from the loss of multiple disks. Refer to the **vxdg**(1M) *nconfig* parameter for more information.

configlen=*length*
The size to reserve for each copy of the configuration stored on the disk. The default size will be based on the size of the private area and the number of configuration copies requested, and will leave some space free for uses other than the configuration copies.

nlogs=*count*
The number of log regions to allocate on the disk. Logs regions are used for storing any plex detaches that happen within the disk group. This number defaults to 1. The Volume Manager will automatically enable and disable the config copy. It will maintain a level of redunancy in configuration copies that will allow the configuration to be recovered from the loss of multiple disks. Refer to the **vxdg**(1M) *nlog* parameter for more information.

loglen=*length*
The size to reserve in the private region for each log region. This size limits the number of kernel-initiated detach operations that can be logged against the disk group. The default is about 15% of the size of the configuration copies. It is advised that the log sizes be kept as 15% of the configuration copy size.

AUTO-CONFIGURED DISKS

On some systems, the Volume Manager can ask the operating system for a list of known disk device addresses. On such systems, some device addresses will be auto-configured into the **rootdg** disk group when **vxconfigd** is started. Auto-configured disks will always be of type **sliced**, with default attributes.

Auto-configured devices can be removed, if necessary, using **vxdisk rm**. When removed, explicitly defined devices can be defined to override any auto-configured devices. When the system reboots, no auto-configured disk devices will be added to the **rootdg** disk group that would share a disk with an explicitly configured disk device.

Auto-configured devices can be disabled and reenabled using the **offline** and **online** operations. However, the **offline** state is not stored persistently. If you need to persistently offline a device at a particular address, you will need to convert the address to use an explicit device record. To do this, remove the auto-configured device, and use **vxdisk define** to create an explicitly configured device.

SEE ALSO

vxintro(1M), **vxconfigd**(1M), **vxdg**(1M), **vxvol**(1M).

NAME

vxdiskadd – add a disk for use with the Volume Manager

SYNOPSIS

vxdiskadd [1]
vxdiskadd c*C***t***T***d***D***

DESCRIPTION

The **vxdiskadd** utility will configure the disk for use by the Volume Manager.

The **vxdiskadd** utility is an interactive command which prompts the user for information about the set up of the disk. As part of its operation, it calls the **vxdisksetup** command in order to generate an initial VTOC (Volume Table Of Contents)and to create the basic set of required partitions. All courses of action that do not involve clear failure conditions require prompts from the user, with defaults supplied as appropriate. Help will be made available at every prompt. Entering **?** in response to a prompt will display a context-sensitive help message.

In the first form of the command, where no disk is specified, or the digit **1** is supplied alone, the second integral disk is added. It is assumed to reside at a disk address of **c0t1d0**.

In its other form, the **c***C***t***T***d***D** argument passed to **vxdiskadd** directs the command to the disk being added. The address defines a controller number (*C*), a SCSI target ID (*T*), and a SCSI logical unit number. Each digit is an octal digit. Special devices, such as internal RAM disks, may use different forms for disk access names. All disk access names used with System V Release 4 relate directly to device node names in the **/dev/dsk** and **/dev/rdsk** directories.

vxdiskadd supports the following general classes of actions:

Reconnecting a drive that was temporarily inaccessible.
: This situation is detected automatically, by noting that the specified drive has a disk ID that matches a disk media record with no currently associated physical disk. After reconnection, any stale plexes referring the disk will be reattached, and any stopped volumes referring the disk will be restarted. This reattach action will be performed by calling the **vxrecover** script.

Encapsulation of disks that have existing contents.
: This will be the default action for disks that do not have a valid, existing volume-manager private region, but that do have a valid UNIX partition and a valid on-disk VTOC. Encapsulation will be performed by calling **vxencap**.

Initialization of a disk with reserved regions and partitions.
: Initialization of the disk will be performed by calling **vxdisksetup**.

Adding a disk to an existing disk group.
: This operation can be performed independently of the initialization of the disk drive, and will add the disk to the group so that it's storage will be added to the free space pool in that disk group. The **vxassist** utility may subsequently allocate from that free space.

Creation of new disk groups in which to import a new disk.
: If disks are added that are required to be put into a disk group that does not

exist, then the option of creating the disk group will be offered.

SEE ALSO

vxintro(1M), **vxassist**(1M), **vxdisksetup**(1M), **vxdisk**(1M), **vxdiskadm**(1M).

NAME

vxdiskadm – administrate disks

SYNOPSIS

vxdiskadm

DESCRIPTION

The **vxdiskadm** script presents a menu of possible operations to the user. Based on the selection of operations, the script guides the user through the necessary steps, and prompts for data that must be user-supplied for completion of the operation. This script is intended for users that understand only a limited set of concepts, and also for users that simply wish to use a simple method for doing common operations.

The **vxdiskadm** script is interactive in nature. It is intended to lead the user of the script through the various possible courses of action that might be taken with a disk that is not currently part of an active disk group. Courses of action that do not involve a clear single alternative require prompts to be answered by the user, with defaults supplied as appropriate. Help is made available at every prompt. Entering **?** in response to a prompt displays a context-sensitive help message.

The **vxdiskadm** script presents the following list of operations:

Add or initialize a disk

This menu option prompts for a disk, by device address. It then checks the disk to ensure that there is no information on the disk. If there is, the operation prompts the user to ensure that the operation should proceed. The user is prompted for a disk group (which defaults to **rootdg**). The user is also given the alternative of initializing the disk but leaving it unallocated to a disk group. If a disk group is specified, the user is then asked to name the disk. A default disk name of the form **disk***number* is chosen for the root disk group. If the disk is to be added to another disk group, then the name will be of the form **disk-group***number* such that the name will be unique within all imported disk groups.

The operation proceeds by calling **vxdisksetup** to partition the disk and to install a disk header. If a disk group was specified, the last step is to add the disk to the indicated disk group.

Encapsulate a disk

This operation prompts for a disk, by device address. It then calls **vxencap** to encapsulate partitions on the named disk.

Remove a disk

This operation prompts for a disk, by media name. The disk is checked to ensure that no subdisks reside on the disk. If the disk is in use, then the operation fails with a recommendation to first move all volumes off the disk. If this disk is the last disk in a disk group, the user is prompted for whether the disk group should be removed from the system, as well.

The operation proceeds by calling **vxdg rmdisk** to remove the disk from its disk group. If this is the last disk in its disk group, then **vxdg deport** is used, instead, to remove the disk group from use.

Remove a disk for replacement

This operation prompts for a disk, by media name. The disk is checked for volumes that would lose all mirrors as a result of the operation. If such volumes exist, those volumes are listed and the user is prompted to ensure that the operation should proceed.

The operation proceeds by calling **vxdg -k rmdisk** to dissociate the media record from the physical disk. If some formatted disks are available that have disk headers but no disk group, then the user is prompted for whether one of these disks should be used as a replacement.

Replace a failed or removed disk

This operation prompts for a disk media name. The named media record must be dissociated from a disk. If the media record is not in the **removed** state, then unused disks are scanned for matching disk IDs. If a disk with a matching disk ID is found, the user is prompted for whether that disk should be reattached.

If a matching disk is not used, the user is prompted for a new disk, by device address. If the named replacement disk has a valid disk header, but is not allocated to a disk group, the user is prompted for whether the disk should be reinitialized. If the named replacement disk is listed as allocated to a disk group or to another host, then the user is prompted to ensure that the operation should proceed.

If the device is to be initialized, **vxdisksetup** is called to partition it and to create the disk header.

Given an initialized disk, the operation proceeds by replacing the disk in a disk group with **vxdg -k adddisk**.

Mirror volumes on a disk

This menu option prompts for a disk, by media name. It then prompts for a destination disk within the same disk group, also by media name. Specifying no destination disks indicates that any disk is suitable. The operation proceeds by calling **vxmirror** to mirror the volumes.

Mirroring volumes from the boot disk will produce a disk that can be used as an alternate boot disk. This will be done by calling the **vxrootmir** command.

Move volumes from a disk

This menu option prompts for a disk, by media name. It then prompts for a possible list of destination disks, also be media name. Specifying no destination disks indicates that any disk is suitable. The operation proceeds by calling **vxevac** to move subdisks off the disk.

Enable access to (import) a disk group

This menu option prompts for a disk, by device address. The operation proceeds by calling **vxdg import** to import the disk group stored on that disk.

Disable access to (deport) a disk group

This menu option prompts for a disk group name. The prompt display lists alternate disk groups and the disks (media name and access name) that they contain. The operation proceeds by calling **vxdg deport**.

Enable (online) a disk device
: This option prompts for a disk device. The prompt display will allow for a display of disks on the system. The operation will only function for disks currently in an offline state. It will then proceed to make the disk accessible.

Disable (offline) a disk device
: This option prompts for a disk device. The prompt display will allow for a display of disks on the system. The operation will only function for disks currently in an online state but not part of any disk group. It will then proceed to mark the disk as offline such that the Volume Manager will make no further attempt at accessing the disk.

Mark a disk as a hot spare
: This option sets up a disk to be used as a hot spare device for it's disk group. A hot spare disk can be used to automatically replace a disk that has failed. No space can be used on a disk that is marked as a hot spare.

Turn off the hot spare flag for a disk
: This option removes a disk from those that can be used as a hot spare and returns its space to the general pool of available space.

Add or initialize all disks on a controller
: This option allows the user to add or initialize all disks on a controller. This option is provided to avoid having to answer redundant questions about the disposition of disks when many disks are being added to the system.

SEE ALSO

vxintro(1M), **vxdg**(1M), **vxdisk**(1M), **vxdisk**(1M), **vxdisksetup**(1M), **vxrootmir(1M).**

NAME

vxdisksetup – set up a disk for use with the Volume Manager

SYNOPSIS

/etc/vx/bin/vxdisksetup [**–nie**] *diskaddress* [*attribute ...*]

/etc/vx/bin/vxdiskunsetup [**–C**] *diskaddress ...*

DESCRIPTION

The **vxdisksetup** command is normally called by the **vxdiskadd** command as part of initial disk configuration. It should not normally be necessary to call this command directly. The **vxdisksetup** command will configure the disk for use by the Volume Manager. **vxdisksetup** will create two partitions on the specified disk, a volume manager private configuration region (called the private region partition), and a partition used for allocating subdisks (called the public region partition). These two partitions have tags that identify them as appropriate to the Volume Manager.

The **vxdiskunsetup** command undoes the configuration setup by **vxdisksetup** and makes the specified disks unusable by the Volume Manager. It can be applied only to disks that are not in active use within an imported disk group.

The *diskaddress* argument passed to **vxdisksetup** or **vxdiskunsetup** directs the command to the disk being operated upon. The address should be provided in the form **c***C***t***T***T***d***D***D*, which define a controller number (*C*), a SCSI target ID (*T*), and a SCSI logical unit number (*D* normally 0). Each number is specified in decimal and can be multiple digits. The name must reference a valid disk with partition devices under the **/dev/rdsk** directory.

vxdisksetup takes the following options:

–n Skip surface analysis of the disk. Surface analysis is normally performed by default on systems where this is applicable. This option is ignored in Solaris (surface analysis is never performed by **vxdisksetup**).

–i Normally, **vxdisksetup** simply creates partitions for use with the Volume Manager. If **–i** is specified, then a disk header will be written to the disk, making the disk directly usable as, for example, a new disk a disk group.

–e Normally, on Solaris, the private region is allocated at the beginning of the disk. This is done so that the VTOC, which is at the beginning of cylinder 0, can be left out of the partition containing subdisks. This makes it more convenient for VxVM to create subdisks that are always on cylinder boundaries. If the **–e** option is specified (this can also be specified as the attribute **privoffset=-0**, then the private region partition is allocated at the end of the disk, and the public region partition is (by default) allocated at the beginning, with an overlap of the VTOC. VxVM will adjust subdisk offsets by 1 block, in that case, so that they will always avoid the VTOC case.

Attributes can be specified to affect the layout strategy used by **vxdisksetup**. Attributes that can be specified are:

publen=*length*

This attribute represents the length of the public region partition of the disk. This defaults to the size of the disk minus the private area on the disk.

privlen=*length*
This attribute represents the length of the private region partition of the disk. The default for this area is 1024 sectors.

puboffset=*offset*
This attribute sets the offset on the disk where the public region partition will start. This defaults to the end of the private region partition, unless the private region partition is moved from the beginning of the disk, in which case the public region offset defaults to following the private region partition.

privoffset=[–]*offset*
This attribute indicates the sector offset of the private area on the disk. The default offset for the private area is at the beginning of the disk. The **–e** option can be used to relocate the private region to the end of the disk. A negative offset can be specified to relocate the private region at an offset relative to the end of the disk.

noconfig
This attribute causes no kernel logs or configuration databases to be set up on the disk. The size of the default private region partition will be adjusted to be 80 blocks, which is thie minimum allowed private region size. This attribute is ignored unless the **–i** option was supplied on the command line.

config This attribute is the converse of the **noconfig** attribute and is the default. This attribute is ignored unless the **–i** option was supplied on the command line.

For Solaris, all lengths and offsets are rounded up to cylinder boundaries, due to restrictions on the layout of partitions.

The **vxdiskunsetup** command operates by removing the partitions that were created by the last invocation of **vxdisksetup** on the given disks. After this operation, and those disks will be converted from the **online** to the **error** state (as displayed by **vxdisk list**). The command refuses to operate on disks that are active members of imported an imported disk group.

Normally, **vxdiskunsetup** also refuses to operate on disks that appear to be imported by some other host (for example, a host that shares access to the disk). In this case, the **–C** option can be specified to force de-partioning of the disk, effectively removing the host locks that were detected by **vxdiskunsetup**.

SEE ALSO

vxintro(1M), **vxdisk**(1M), **vxdiskadd**(1M).

NAME

vxedit – create, remove, and modify Volume Manager records

SYNOPSIS

vxedit [**–g** *diskgroup*] [**–e** *pattern*] [**–vpsdGrf**] **set** *attribute***=***value* ... [*name* ...]
vxedit [**–g** *diskgroup*] [**–e** *pattern*] [**–vpsdGr**] **cc** /*search*/*replace*/[**gp**] [*name* ...]
vxedit [**–g** *diskgroup*] [**–vpsdrf**] **rm** *name* ...
vxedit [**–g** *diskgroup*] [**–vpsd**] **rename** *oldname newname*

DESCRIPTION

The **vxedit** utility sets and changes various attributes for Volume Manager configuration records that do not depend upon volume usage types. See **vxvol**(1M) for operations that can set attributes that are dependent upon usage types. In particular, setting the length and logging type for a volume requires use of the **vxvol set** operation.

Each invocation can be applied to only one disk group at a time, due to internal implementation constraints. Any *name* or *oldname* operands will be used as record names to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). If no *name* or *oldname* operands are given, then the disk group defaults to **rootdg**. A specific disk group can be forced with **–g** *diskgroup*.

These are the recognized operation keywords:

set Set a field within a volume, plex, subdisk, disk media, or disk group record in the selected disk group. The records to be changed are those that match the pattern specified with the **–e** *pattern* option and those specified by the *name* operands.

The *attribute* names specify the field to set within the selected records. More than one attribute can be specified in a single invocation. The operands that indicate attribute settings end at the first operand that does not contain an equal sign. An operand of –– can be used to separate the attribute list from record names, even if the first record name contains an equal sign.

If the **–r** option is given, the operation is applied recursively to records associated with the selected records (to plexes and subdisks for selected volume records, and to subdisks for selected plex records). Recursion applies regardless of **–p** and **–s** options.

The following values can be set for all record types except for disk access records:

comment or **c**
Set the comment string for the selected records to the given value. The comment string cannot be longer than 40 characters and cannot contain a newline character.

tutil0, **tutil1**, **tutil2**
Set one of the non-persistent (temporary) utility fields in the record.

putil0, **putil1**, **putil2**
Set one of the non-persistent (temporary) utility fields in the record. The six utility fields cannot be longer than 14 characters and cannot contain a newline character. The **tutil0** and **putil0** fields are reserved for use by the the base Volume Manager utility set, and by usage types. The **tutil1** and **putil1** fields are reserved for use by higher-level utilities. The **tutil2**

and **putil2** fields are reserved for any desired use by system administrators.

The **putil0** field for a plex and subdisk record can be set to prevent utilities from associating the record to a volume or plex. This is a convenient means for reserving a plex, or for reserving a region of disk space (a subdisk).

Changing a non-empty **putil0** or **tutil0** field requires use of **–f**. Setting these fields for dissociated plex and subdisk records is generally not a problem. However, for an associated plex or subdisk that is associated (through a plex) with a volume, setting either of these fields can be dangerous, as it can affect the operation of usage types that expect to manage these fields themselves during an operation.

The following attributes can only be set for volume records:

fstype If a volume contains a file system, **fstype** can be used to indicate what type of file system it is. Under most circumstances, if a file system type is not specified for a volume, the Volume Manager will determine the usage type by running the **fstyp** program (see **fstyp**(1M)). However, it is preferable to set **fstype**, to avoid problems when the **fstyp** program returns ambiguous results.

writecopy

Set or clear (set to *on* or *off*) a volume policy that affects consistency of data written to a volume when dirty-region-logging is in effect on the volume. When the operating system hands off a write request to the volume driver, the operating system may continue to change the memory that is being written to disk. The volume manager cannot detect that the memory is changing, so it can inadvertently leave plexes with inconsistent contents.

This is not normally a problem, because the operating system ensures that any such modified memory is rewritten to the volume before the volume is closed, such as by a clean system shutdown. However, if the system crashes, plexes may be inconsistent. Since the dirty-region-logging feature prevents recovery of the entire volume, it may not ensure that plexes are entirely consistent.

Setting the **writecopy** flag, often causes the Volume Manager to copy the data for a write request to a new section of memory before writing it to disk. Because the write is done from the copied memory, it can't change and so the data written to each plex is guaranteed to be the same.

writeback

Set (**on**) or clear (**off**) a volume policy that affects recovery after read failures on a mirrored volume. If the **writeback** flag is set (which is normally the default), then a read failure for a plex will cause data to be read from an alternate plex and then written back to the plex that got the read failure. This will usually fix the error. Only if the writeback fails will the

plex be detached for having an unrecoverable I/O failure.

If this flag is clear, then data from an alternate plex will be read to satisfy the volume read operation, but the failing plex will be detached with no action taken to try to fix the problem.

There is seldom (if ever) a reason to turn off this feature.

user Set the user that owns a volume record to the user given as the attribute value. The attribute value can be either a login name from the **/etc/passwd** database, or a numeric user ID.

group Set the group that owns a volume record to the group given as the attribute value. The attribute value can be either a group name from the group database, or a numeric group ID.

mode Set the access permissions for the volume to the permission mode given in the attribute value. The attribute value can be a symbolic permission mode or an octal mode. The format is compatible with permission modes as used by the **chmod** utility (see **chmod**(1)).

The following special attribute can be set for subdisk records:

len Set the length of the subdisk to the given length. The attribute value is a standard Volume Manager length number (see **vxintro**(1M)). The length of a subdisk can be changed only if the subdisk is dissociated. The length of a subdisk cannot be increased to the point where it would extend past the end of the disk, or to where it would overlap a reserved disk region or another subdisk.

The following special attribute can be set for disk media records:

reserve Set (**on**) or clear (**off**) the disk reservation flag. If the reserve flag is set for a disk, then **vxassist** will not allocate a subdisk on that disk unless the disk is specified on the **vxassist** command line.

spare Set (**on**) or clear (**off**) the disk hot spare flag. If the hot spare flag is set for a disk, then **vxassist** will not allocate a subdisk on that disk unless the disk is specified on the **vxassist** command line. A disk with spare flag on will be used as hot spare when a disk fails.

cc Change a comment using a search-replacement specification similar to that used by **sed**(1) in volume, plex, subdisk, disk media, or disk group records within the selected disk group. The records to be changed are those that match the pattern specified with **–e** *pattern* option and those specified by the *name* operands. See **vxintro**(1M) for a description of Volume Manager search patterns. If no search pattern is specified with **–e**, and no *name* operands are given, then the change is made to all records whose comment field matches the *search* regular expression.

The *search* string is a regular expression, in the form accepted by the function **regcmp**(3G). This regular expression is used to determine which substring of the comment field is to be changed. The *replace* string represents the new string to use as a replacement for the matched part of the comment.

An occurrence of **&** in the *replace* string will be replaced by the substring of the comment matched by the regular expression. An occurrence of *n* in the replace string, where *n* is a single digit between 1 and 9, will be replaced by the substring matched by a parenthetical section of the regular expression; the regular expression is followed by **$***n*.

The **/** character following the *replace* string is optional. If the **/** is given, then it can be followed by the letters **g** or **p**, or both. If a **g** is given, then all matches in a comment are replaced, rather than just the first match. If the letter **p** is given, then the resulting comment strings are written to the standard output, immediately preceded (on the same line) by the name of the record.

If the **–r** option is given, the operation is applied recursively to records associated with the selected records (to plexes and subdisks for selected volume records, and to subdisks for selected plex records). Recursion (when selected) applies regardless of the **–p**, and **–s** options.

Each record to be changed is changed only once, even if the record could be matched several times through combinations of *name* arguments, search patterns, and the **–r** option.

For example, the following command changes all subdisk comments that begin with ‘‘Henry’’ and a second word beginning with an uppercase letter to begin with ‘‘Frank’’ and the same second word:

vxedit –s cc ’/^Henry ([A-Z])$1/Frank \1/p’

This command also lists the resulting comment fields.

rm Remove volume, plex, or subdisk records from the selected disk group. Disk media records can be removed with **vxdg rmdisk**. Disk access records can be removed with **vxdisk rm**.

Removing a subdisk requires that the subdisk be dissociated. Removing a plex requires that the plex be dissociated and that it have no associated subdisks. Removing a volume requires that it have no associated plexes. The **–r** option can be specified to remove a volume and all plex and subdisk records associated with it, or to remove a plex and all subdisk records associated with it. Even with **–r**, a named plex or subdisk cannot be associated with a volume or plex, respectively.

The **–f** option is required to remove an enabled volume. A volume cannot be removed, even with **–f**, if the corresponding volume block or raw device, or if any of the volume’s plex devices, is open or mounted. Likewise, a plex cannot be removed if the corresponding plex device is open.

rename

Change the name of a volume, plex, subdisk, or disk media record from *oldname* to *newname*. A record cannot be renamed if the **tutil0** field is set, which indicates that an operation is in progress that involves the record.

OPTIONS

The following options are recognized:

–g *diskgroup* Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *name* and *oldname* operands.

–f Force an operation that the Volume Manager considers potentially dangerous or is not a normal operation for the command. This enables a limited set of operations that would otherwise be disallowed. Some operations may be disallowed even with this flag. The **vxedit** operations that are allowed with this flag are changing a non-empty **tutil0** or **putil0** field, and removing enabled volumes.

–v, **–p**, **–s**, **–d**, **–G** Select only volume, plex, subdisk, disk media, or disk group records, respectively. If more than one of these options are specified, then records of any of the indicated types may be selected.

–r Operate, recursively, on records associated with the selected records. For selected volume records, this affects associated plex and subdisk records. For selected plex records, this affects associated subdisk records.

–e *pattern* Use a Volume Manager configuration search expression to select records from the selected disk group configuration. Search patterns are currently limited to a selection of volume, plex, and subdisk records.

EXIT CODES

The **vxedit** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility.

See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

chmod(1), **regcmp**(3G), **vxintro**(1M), **vxmake**(1M), **vxmend**(1M), **vxvol**(1M).

NAME

vxencap – encapsulate partitions on a new disk

SYNOPSIS

/etc/vx/bin/vxencap [**–g** *diskgroup*] [**–Nc**] [*medianame*=]*accessname*

DESCRIPTION

The **vxencap** script identifies any partitions on the specified disk that could be used for file systems or special areas such as swap devices, and then generates volumes to cover those areas on the disk. This process is known as *encapsulation*. In the case of partitions that contain file systems, the entries in **/etc/vfstab** will be modified to use the new volume devices.

vxencap is generally called from the **vxdiskadm** menus or else is called at installation time from the **vxinstall** command.

Options that can be specified to **vxencap** are:

–g *diskgroup*
: Place the encapsulated disk into the disk group specified by disk group ID or disk group name.

–N
: Specify that the changes that have been logged to be made during the next reboot should not be committed. In the usual case of operation of this command, any system failure before the reboot will cause the operations to be completed on the next reboot. With the **–N** flag specified, the changes will be left uncompleted and the operation aborted should the system fail at any point before the successful configuration change. This is accomplished by not creating the usual **/etc/vx/reconfig.d/state.d/reconfig** commit file.

–c
: This option tells the command to create a disk group if the one specified does not exist. This option is only useful if a disk group is also supplied using the **-g** option.

SEE ALSO

vxdiskadm(1M), **vxintro**(1M), *VERITAS Volume Manager (VxVM) Installation Guide.*

NAME

vxevac – evacuate all volumes from a disk

SYNOPSIS

/etc/vx/bin/vxevac [**–g** *diskgroup*] *medianame* [*new_medianame* ...]

DESCRIPTION

The **vxevac** script moves subdisks off the specified disk (*medianame*) to the specified destination disks (*new medianame* ...). If no *new medianame* operands are specified, then any non-volatile, non-reserved disks can be used as destination disks. Subdisks that are part of unmirrored striped plexes will be moved by moving the entire plex to a new location.

This operation presumes that sufficient space exists elsewhere in the disk group for it to complete the operation. If the process runs out of space, then not all of the volumes on the disk may be evacuated.

This script is usually called from the **vxdiskadm** menus.

SEE ALSO

vxintro(1M), **vxdiskadm**(1M).

NAME

vxinfo – print accessibility and usability of volumes

SYNOPSIS

vxinfo [**–Vp**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [*volume* ...]

DESCRIPTION

The **vxinfo** utility reports a usage-type-dependent condition on one or more volumes in a disk group. A report for each volume specified by the *volume* operand is written to the standard output. If no *volume* operands are given, then a volume condition report is provided for each volume in the selected disk group.

Each invocation can be applied to only one disk group at a time, due to internal implementation constraints. Any *volume* operands will be used to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). A specific disk group can be forced with **–g** *diskgroup*.

OPTION

The following options are recognized:

–V Write a list of utilities that would be called from **vxinfo**, along with the arguments that would be passed. The **–V** performs a ''mock run'' so the utilities are not actually called.

–p Report the name and condition of each plex in each reported volume.

–U *usetype* Specify the usage type for the operation. If no *volume* operands are specified, then the output is restricted to volumes with this usage type. If *volume* operands are specified, then this will result in a failure message for all named volumes that do not have the indicated usage type.

–g *diskgroup* Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *volume* operands. If no *volume* operands are specified, the disk group defaults to **rootdg**.

–o *useopt* Pass in usage-type-specific options to the operation.

VOLUME CONDITIONS

The volume condition is a usage-type-dependent summary of the state of a volume. This condition is derived from the volume's kernel-enabled state and the usage-type-dependent states of the volume's plexes.

The **vxinfo** utility reports the following conditions for volumes:

Startable A **vxvol startall** operation would likely succeed in starting the volume.

Unstartable The volume is not started and either is not correctly configured or doesn't meet the prerequisites for automatic startup (with **volume startup**) because of errors or other conditions.

Started The volume has been started and can be used.

Started Unusable The volume has been started but is not operationally accessible. This condition may result from errors that have occurred since the volume was started, or may be a result of administrative actions, such as **vxdg -k rmdisk**.

OUTPUT FORMAT

Summary reports for each volume are printed in one-line output records. The volume output line consists of blank-separated fields for the volume name, volume condition, and the type of the volume. The following example shows the volume summary:

```
bigvol      fsgen   Startable
vol2        fsgen   Startable
brokenvol   gen     Unstartable
```

The output format for each plex is a one-line output record consisting of blank-separated fields for the plex name and the plex condition, as a usage-type-dependent string. The plex records are indented from the volume records, as the following example shows:

```
vol  bigvol       fsgen   Startable
plex bigvol-01    ACTIVE
vol  vol2         fsgen   Startable
plex vol2-01      ACTIVE
vol  brokenvol    gen     Unstartable
```

FSGEN AND GEN USAGE-TYPES

The **fsgen** and **gen** usage types provide identical semantics for the **vxinfo** utility. The **fsgen** and **gen** usage types do not support any options passed in with **–o**.

Plex conditions (reported with **–p**) can be one of the following:

NODAREC No physical disk was found for one of the subdisks in the plex. This implies either that the physical disk failed, making it unrecognizable, or that the physical disk is no longer attached through a known access path.

REMOVED A physical disk used by one of the subdisks in the plex was removed through administrative action with **vxdg -k rmdisk**.

IOFAIL The plex was detached from use as a result of an uncorrectable I/O failure on one of the subdisks in the plex.

STALE The plex does not contain valid data, either as a result of a disk replacement affecting one of the subdisks in the plex, or as a result of an administrative action on the plex such as **vxplex det**.

CLEAN The plex contains valid data and the volume was stopped cleanly.

ACTIVE Either the volume is started and the plex is enabled, or the volume was not stopped cleanly and the plex was valid when the volume was stopped.

OFFLINE The plex was disabled using the **vxmend off** operation.

EMPTY The plex is part of a volume that has not yet been initialized.

TEMP The plex is associated temporarily as part of a current operation, such as **vxplex cp** or **vxplex att**. A system reboot or manual starting of a volume will dissociate the plex.

TEMPRM The plex was created for temporary use by a current operation. A system reboot or manual starting of a volume will remove the plex.

TEMPRMSD The plex and its subdisks were created for temporary use by a current operation. A system reboot or manual starting of the volume will remove the plex and all of its subdisks.

SNAPATT The plex is being attached as part of a backup operation by the **vxassist snapstart** operation. When the attach is complete, the condition will change to **SNAPDONE**. A system reboot or manual starting of the volume will remove the plex and all of its subdisks.

SNAPDONE A **vxassist snapstart** operation completed the process of attaching the plex. It is a candidate for selection by the **vxassist snapshot** operation. A system reboot or manual starting of the volume will remove the plex and all of its subdisks.

SNAPTMP The plex is being attached as part of a backup operation by the **vxplex snapstart** operation. When the attach is complete, the condition will change to **SNAPDIS**. A system reboot or manual starting of the volume will dissociate the plex.

SNAPDIS A **vxassist snapstart** operation completed the process of attaching the plex. It is a candidate for selection by the **vxplex snapshot** operation. A system reboot or manual starting of the volume will dissociate the plex.

Volume conditions for these usage types are reported as follows:

Startable This condition is reported if the volume is not enabled and if any of the plexes have a reported condition of **ACTIVE** or **CLEAN**.

Unstartable This condition is reported if the volume is not enabled, but the volume does not meet the criteria for being **Startable**.

Started This condition is reported if the volume is enabled and at least one of the associated plexes is enabled in read-write mode (which is normal for enabled plexes in the **ACTIVE** and **EMTPY** conditions).

Started Unusable
This condition is reported if the volume is enabled, but the volume does not meet the criteria for being **Started**.

RAID5 USAGE-TYPE

The **raid5** usage type does not support any options passed in with **-o**.

Plexes of RAID-5 volumes can be either data plexes (i.e. RAID-5 plexes) or logging plexes. RAID-5 plex conditions are as follows:

NODAREC No physical disk was found for one of the subdisks in the plex. This implies either that the physical disk failed, making it unrecognizable, or that the physical disk is no longer attached through a known access path.

REMOVED A physical disk used by one of the subdisks in the plex was removed through administrative action with **vxdg -k rmdisk**.

IOFAIL The plex was detached from use as a result of an uncorrectable I / O failure on one of the subdisks in the plex.

CLEAN The plex contains valid data and the volume was stopped cleanly.

ACTIVE Either the volume is started and the plex is enabled, or the volume was not stopped cleanly and the plex was valid when the volume was stopped.

OFFLINE The plex was disabled using the **vxmend off** operation.

RAID-5 plexes can also have these additional conditions:

DEGRADED Due to subdisk failures the plex is in degraded mode. This indicates a loss of data redundancy in the RAID-5 volume and any further failures could cause data loss.

STALEPRTY The parity is not in sync with the data in the plex. This indicates a loss of data redundancy in the RAID-5 volume and any further failures could case data loss.

UNUSABLE This indicates a double-failure occurred within the plex. The plex is unusable due to subdisk failures and / or stale parity.

Logging plexes of RAID-5 volumes can have the following conditions:

NODAREC No physical disk was found for one of the subdisks in the plex. This implies either that the physical disk failed, making it unrecognizable, or that the physical disk is no longer attached through a known access path.

REMOVED A physical disk used by one of the subdisks in the plex was removed through administrative action with **vxdg -k rmdisk**.

IOFAIL The plex was detached from use as a result of an uncorrectable I / O failure on one of the subdisks in the plex.

CLEAN The plex contains valid data and the volume was stopped cleanly.

ACTIVE Either the volume is started and the plex is enabled, or the volume was not stopped cleanly and the plex was valid when the volume was stopped.

OFFLINE The plex was disabled using the **vxmend off** operation.

Logging plexes of RAID5 volumes can also have the following additional condition:

BADLOG
The contents of the plex are not usable as logging data.

Volume conditions for volumes of the **raid5** usage type are the same as for the **fsgen** and **gen** usage types (i.e., **Startable**, **Unstartable**, **Started** and **Started Unusable**). In addition, the following additional conditions can modify those conditions:

Degraded
This condition indicates that the RAID-5 plex of the volume is in degraded mode due to the unavailability of a subdisk in that plex.

Staleprty

This condition indicates that some of the parity in the RAID-5 plex is stale and requires recovery.

FILES

/etc/vx/type/*usetype***/vxinfo**
The utility that performs **vxinfo** operations for a particular volume usage type.

EXIT CODES

The **vxinfo** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility. See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

vxintro(1M), **vxassist**(1M), **vxmend**(1M), **vxplex**(1M), **vxsd**(1M), **vxvol**(1M).

NAME

`vxintro` – Introduction to Volume Manager utilities

SYNOPSIS

vxassist, vxconfigd, vxdctl, vxdg, vxdisk, vxedit, vxinfo, vxiod, vxmake, vxmend, vxplex, vxprint, vxrecover, vxsd, vxstat, vxtrace, vxvol

DESCRIPTION

The Volume Manager utilities provide a shell-level interface used by the system administrator and higher-level applications and scripts to query and manipulate objects that are managed through the Volume Manager (VxVM®).

GLOSSARY

Some of the terms and objects that are used with the Volume Manager are:

volume: A virtual disk device that looks to applications and file systems like a regular disk partition device. Volumes present block and raw device interfaces that are compatible in their use, with disk partition devices. However, a volume is a virtual device that can be mirrored, spanned across disk drives, moved to use different storage, and striped using administrative commands. The configuration of a volume can be changed, using the Volume Manager utilities, without causing disruption to applications or file systems that are using the volume.

plex: A copy of a volume's logical data address space, also sometimes known as a *mirror*. A volume can have up to eight plexes associated with it. Each plex is, at least conceptually, a copy of the volume that is maintained consistently in the presence of volume I/O and reconfigurations. Plexes represent the primary means of configuring storage for a volume. Plexes can have a striped or concatenated organization (layout).

disk: Disks exist as two entities. One is the physical disk on which all data is ultimately stored and which exhibits all the behaviors of the underlying technology. The other is the Volume Manager presentation of disks which, while mapping one-to-one with the physical disks, are just presentations of units from which allocations of storage are made. As an example, a physical disk presents the image of a device with a definable geometry with a definable number of cylinders, heads etc. whereas a Volume Manager disk is simply a unit of allocation with a name and a size.

subdisk: A region of storage allocated on a disk for use with a volume. Subdisks are associated to volumes through plexes. One or more subdisks are layout to form plexes based on the plex layout: striped or concatenated. Subdisks are defined relative to disk media records.

disk media record: A reference to a physical disk, or possibly a disk partition. This record can be thought of as a physical disk identifier for the disk or partition. Disk media records are configuration records that provide a name (known as the disk media name or DM name) that an administrator can use to reference a particular disk independent of its location on the system's various disk controllers. Disk media records reference particular physical disks through a disk ID, which is a unique identifier that is assigned to a disk when it is initialized for use with the Volume Manager.

Operations are provided to set or remove the disk ID stored in a disk media record. Such operations have the effect of removing or replacing disks, with any associated subdisks being removed or replaced along with the disk.

disk access record: A configuration record that defines a pathway to a disk. The disk access records most commonly name a particular controller number, target ID, and logical unit number. The list of all disk access records stored in a system is used to find all disks attached to the system. Disk access records do not identify particular physical disks.

Through the use of disk IDs, the Volume Manager allows disks to be moved between controllers, or to different locations on a controller. When a disk is moved, a different disk access record will be used when accessing the disk, although the disk media record will continue to track the actual physical disk.

On some systems, the Volume Manager will build a list of disk access records automatically, based on the list of all devices attached to the system. On these systems, it is not necessary to define disk access records explicitly. On other systems, disk access records must be defined explicitly with the **vxdisk define** operation. Specialty disks (such as RAM disks or floppy disks) are likely to require explicit **vxdisk define** operations on all systems.

Disk access records are identified by their disk access names (also known as DA names).

disk group: A group of disks that share a common configuration. A configuration consists of a set of records describing objects including disks, volumes, plexes, and subdisks that are associated with one particular disk group. Each disk group has an administrator-assigned name that can be used by the administrator to reference that disk group. Each disk group has an internally defined unique disk group ID, which is used to differentiate two disk groups with the same administrator-assigned name.

Disk groups provide a method to partition the configuration database, so that the database size is not too large and so that database modifications do not affect too many drives. They also allow the Volume Manager to operate with groups of physical disk media that can be moved between systems.

Disks and disk groups have a circular relationship: disk groups are formed from disks, and disk group configurations are stored on disks. All disks in a disk group are stamped with a disk group ID, which is a unique identifier for naming disk groups. Some or all disks in a disk group also store copies of the configuration of the disk group.

disk group configuration: A disk group configuration is a small database that contains all volume, plex, subdisk, and disk media records. These configurations are replicated onto some or all disks in the disk group usually with one copy on each disk. Because these databases are stored within disk groups, record associations cannot span disk groups. Thus, a subdisk defined on a disk in one disk group cannot be associated with a volume in another disk group.

root disk group: Each system requires one special disk group, named **rootdg**. This group is generally the default for most utilities. In addition to defining the regular disk group information, the configuration for the root disk group contains local information that is specific to a disk group and that is not intended to be movable between systems.

private region: Disks used by the Volume Manager contain two special regions: a private region and a public region. Usually, each region is formed from a complete partition of the disk; however, the private and public regions can be allocated from the same partition.

The private region of a disk contains various on-disk structures that are used by the Volume Manager for various internal purposes. Each private region begins with a disk header which identifies the disk and its disk group. Private regions can also contain copies of a disk group's configuration, and copies of the disk group's kernel log.

public region: The public region of a disk is the space reserved for allocating subdisks. Subdisks are defined with offsets that are relative to the beginning of the public region of a particular disk. Only one contiguous region of disk can form the public region for a particular disk.

kernel log: A log kept in the private region on the disk and that is written by the Volume Manager kernel. The log contains records describing the state of volumes in the disk group. This log provides a mechanism for the kernel to persistently register state changes so that **vxconfigd** can be guaranteed to detect the state changes even in the event of a system failure.

disk header: A block stored in a private region of a disk and that defines several properties of the disk. The disk header defines the size of the private region, the location and size of the public region, the unique disk ID for the disk, the disk group ID and disk group name (if the disk is currently associated with a disk group), and the host ID for a host that has exclusive use of the disk.

disk ID: A 64 byte universally unique identifier that is assigned to a physical disk when its private region is initialized with the **vxdisk init** operation. The disk ID is stored in the disk media record so that the physical disk can be related to the disk media record at system startup.

disk group ID: A 64 byte universally unique identifier that is assigned to a disk group when the disk group is created with **vxdg init**. This identifier is in addition to the disk group name, which is assigned by the administrator. The disk group ID is used to check for disk groups that have the same administrator-assigned name but are actually distinct.

host ID: A name, usually assigned by the administrator, that identifies a particular host. Host IDs are used to assign ownership to particular physical disks. When a disk is part of a disk group that is in active use by a particular host, the disk is stamped with that host's host ID. If another system attempts to access the disk, it will detect that the disk has a non-matching host ID and will disallow access until the first system discontinues use of the disk. To allow for system failures that do not clear the

host ID, the **vxdisk clearimport** operation can be used to clear the host ID stored on a disk.

If a disk is a member of a disk group and has a host ID that matches a particular host, then that host will import the disk group as part of system startup.

striped plex: A plex that scatters data evenly across each of its associated subdisks. A plex has a characteristic number of stripe columns (represented by the number of associated subdisks) and a characteristic stripe width. The stripe width defines how data with a particular address is allocated to one of the associated subdisks. Given a stripe width of 128 blocks, and two stripe columns, the first group of 128 blocks would be allocated to the first subdisk, the second group of 128 blocks would be allocated to the second subdisk, the third group to the first subdisk, again, and so on.

concatenated plex: A plex whose subdisks are associated at specific offsets within the address range of the plex, and extend in the plex address range for the length of the subdisk. This layout allows regions of one or more disks to create a plex, rather then a single big region.

volboot file: The **volboot** file is a special file (usually stored in **/etc/vx/volboot**) that is used to bootstrap the root disk group and to define a system's host ID. In addition to a host ID, the **volboot** file contains a list of disk access records. On system startup, this list of disks is scanned to find a disk that is a member of the **rootdg** disk group and that is stamped with this system's host ID. When such a disk is found, its configuration is read and is used to get a more complete list of disk access records that are used as a second-stage bootstrap of the root disk group, and to locate all other disk groups.

plex consistency: If the plexes of a volume contain different data, then the plexes are said to be inconsistent. This is only a problem if the Volume Manager is unaware of the inconsistencies, as the volume can return differing results for consecutive reads. Plex inconsistency is a serious compromise of data integrity. This inconsistency can be caused by write operations that start around the time of a system failure, if parts of the write complete on one plex but not the other. Plexes can also be inconsistent after creation of a mirrored volume, if the plexes are not first synchronized to contain the same data. An important part of Volume Manager operation is ensuring that consistent data is returned to any application that reads a volume. This may require that plex consistency of a volume be ''recovered'' by copying data between plexes so that they have the same contents. Alternatively, the volume can be put into a state such that reads from one plex are automatically written back to the other plexes, thus making the data consistent for that volume offset.

CONVENTIONS

A number of conventions are used throughout much of the Volume Manager to provide a degree of similarity between the various operations. The following is a list of such conventions:

Utility syntax

Most utilities in the Volume Manager provide more than one operation, with operations grouped into utilities primarily by object type. Utilities that provide multiple operations are typically invoked with the following form:

utility [*options*] *keyword* [*operands*]

Here, *utility* is the name of the utility and *keyword* is a name that identifies the specific operation to perform. Any options that are introduced in the standard –*letter* form precede the operation keyword. This is in keeping with standard System V utility syntax, which provides for a set of options as the first arguments to a command, followed by any non-option operands. The *keyword* is considered an operand under standard System V syntax.

To aid in normal use, all of the utilities provide an extended usage message that lists all the options and operation keywords supported by the utility. For utilities that are keyword-based, this extended usage message can be displayed using a keyword of **help**. Utilities that use operands for purposes other than operation selection provide a reserved option of **–H** to display the extended usage information. These extended messages cannot replace the user documentation, but can serve as reminders.

Standard length numbers

Many basic properties of objects that are managed by the Volume Manager require specification of lengths, either as a pure object length or as an offset relative to some other object. The Volume Manager supports volume lengths up to 2,147,483,647 disk sectors (one terabyte on most systems). Typing such large numbers, or even much smaller numbers, can be annoying. The Volume Manager provides a uniform syntax for representing such numbers which uses suffixes to provide convenient multipliers. Numbers can be specified in decimal, octal or hexadecimal. Also, numbers can be specified as a sum of several numbers, as a convenience to avoid using a calculator.

A hexadecimal (base 16) number is introduced using a prefix of **0x**. For example, **0xfff** is the same as decimal 4091. An octal (base 8) number is introduced using a prefix of **0**. For example, **0177777** is the same as decimal 65535.

A number can be followed by a suffix character to indicate a multiplier for the number. A length number with no suffix character represents a count of standard disk sectors. The length of a standard disk sector can vary between systems. It is commonly 512 bytes. On systems where disks can have different sector sizes, one of the sectors sizes will be chosen as the ‘‘standard’’ size. Supported suffix characters are:

b multiply the length by 512 bytes (blocks)
s multiply the length by the standard sectors size (default)
k multiply the length by 1024 bytes (Kilobytes)
m multiply the length by 1,048,576 (1024K) bytes (Megabytes)
g multiply the length by 1,073,741,824 (1024M) bytes (Gigabytes)

Numbers are represented internally as an integer number of sectors. As a result, if the standard disk sectors size is larger than 512 bytes, numbers can be specified that will need to be rounded to a sector. Rounding is always done to the next lowest, not the nearest, multiple of the sector size.

Since the letter **b** is a valid hexadecimal character, there is a special case for the **b** suffix where a single blank character can separate a number from the **b** suffix character. Use of a blank within a number, when invoking commands from the shell, usually requires quoting the number. For example:

vxassist make vol01 "0x1000 b"

Numbers can be added or subtracted by separating two or more numbers by a plus or minus sign, respectively. A plus sign is optional. As an example, the largest allowed number that can be represented on a system with a 512 byte sector size can be entered as:

1023g+1023m+1023k+1

Note that **1024g-1** cannot be used, because the implementation cannot handle the intermediate representation of **1024g** (which is greater than the largest number that can be represented) internally. However, **2g-1** can be used to represent the largest volume size that can be used with most file systems.

The Volume Manager always reports length numbers as a simple count of sectors, with no suffix character.

Case is not important in length numbers. Hexadecimal numbers and suffix characters can be specified using any reasonable combination of upper- and lower-case letters.

Disk group selection

Most commands operate upon only one disk group per invocation. Each disk group has a separate configuration from every other disk group and it is possible for two disk groups to contain two objects that have the same name. This can happen, in particular, if a disk group is moved from one system to another. However, most utilities make no attempt to ensure that names between disk groups are unique, so name collisions can occur anyway.

System administrators who endeavor to avoid name collisions should be able to use most of the utilities without having to specify disk groups except when creating objects. Administrators cannot use single-command invocations that reference objects in more than one disk group, but disk groups will be selected automatically, based on objects specified in the command.

The standard rules that most commands use for selecting the disk group for a command are as follows:

- Given a particular set of object names specified on the command, look for the disk group of each object. If all objects are in the same disk group, use that disk group. If any named object is not unique between all disk groups, and if the one of those object names is not in the **rootdg** disk group, then fail.
- To force use of a particular disk group, use **–g** *diskgroup* to indicate the group. Non-unique names do not cause errors when a disk group is specified explicitly. The *diskgroup* specification is either a disk group ID or a disk group name.
- A special case is provided for the **rootdg** disk group. Any set of objects in the **rootdg** disk group can be specified without specifying **–g rootdg**, even if the name is used in another disk group.

If a set of object names is given on the command line, and if some are unique but some are not unique, then the command will still fail according to the rules listed above. Just because a combination of objects could be used to disambiguate the disk group does not mean that a utility will do so.

RECORD TYPES

Disk group configurations contain six types of records: volume records, plex records, subdisk records, disk media records, disk group records, and disk access records. Disk access records are specific to the root disk group and are stored in configurations only because there is no other convenient place to store them; otherwise, they are logically separate from all disk groups. Since they are specific and meaningful to the local system only, the logical place for their storage is the **rootdg** since that is the only disk group guaranteed to exist on the system.

Disk group records

Disk group records define several different types of names for a disk group. The different types of names are:

real name — This is the name of the disk group, as the name is defined on disk. This name is stored in the disk group configuration, and is also stored in the disk headers of all disks in the disk group.

alias name — This is the standard name that the system uses when referencing the disk group. References to the disk group name usually mean the alias name. Volume and plex device directories are structured into subdirectories based on the disk group alias name. Typically, the disk group's alias name and real name are identical. A local alias can be useful for gaining access to a disk group with a name that conflicts with other disk groups in the system, or that conflicts with records in the **rootdg** disk group.

disk group ID — A 64-byte identifier that represents the unique ID of the disk group. All disk groups on all systems should have a different disk group ID, even if they have the same real name. This identifier is stored in the disk headers of all disks in the disk group. It is used to ensure that the Volume Manager does not confuse two disk groups which were created with the same name.

Volume records

Volume records define the characteristics of particular volume devices. The name of a volume record defines the node name used for files in the **/dev/vx/dsk** and **/dev/vx/rdsk** directories. The block device for a particular volume (which can be used as an argument to the **mount** command (see **mount**(1M)) has the path:

/dev/vx/dsk/*groupname*/*volume*

where *groupname* is the name assigned by the administrator to the disk group containing the volume. The raw device for a volume, typically used for application I/O and for issuing I/O control operations (see **ioctl**(2)), has the path:

/dev/vx/rdsk/*groupname*/*volume*

For convenience, volumes assigned to the root disk group are accessible under the **rootdg** subdirectories of **/dev/vx/dsk** and **/dev/vx/rdsk**, but are also under **/dev/vx/dsk/***volume* and **/dev/vx/rdsk/***volume*.

Reads to a volume device are directed to one of the read-write or read-only plexes associated with the volume. Writes to the volume are directed to all of the enabled read-write and write-only plexes associated with the volume.

During a write operation, two plexes of a volume may become out of sync with each other, due to the fact that writes directed to two disks can complete at different times. This is not normally a problem. However, if the system were to crash or lose power during a write operation, the two plexes could have different contents.

Most applications and file systems are not written with the presumption that two separate reads of a device can return different contents without an intervening write operation. Since plexes with different contents could cause such a situation where two read operations of a block return different contents, the Volume Manager expends considerable effort to ensure that this is avoided.

Volumes have the following fundamental attributes:

usage type — Each volume has a usage type, which defines a particular class of rules for operating on the volume, typically based on the expected content of the volume. Several utilities can apply extensions or limitations that apply to volumes with a particular usage type. Four usage types are included with the base release of the Volume Manager: **fsgen**, for use with volumes that contain file systems; **gen**, for use with volumes that are used as swap devices or for other applications that do not use file systems; and special **root** and **swap** usage types which are specifically for use with the root file system volume and the primary swap device.

length — Each volume has a length, which defines the limiting offset of read and write operations. The length is assigned by the administrator, and may or may not match the lengths of the associated plexes.

volume state — Each volume is either enabled, disabled, or detached. When enabled, normal read and write operations are allowed on the volume, and any file system residing on the volume can be mounted, or used in the usual way. When disabled, no access to the volume or any of its associated plexes is allowed. When detached, the plex devices for the volume can be accessed, and some ioctls can be used by utilities to operate on the volume.

usage-type state — Usage types maintain a private state field related to the volume that relate to operations that have been performed on the volume, or to failure conditions that have been encountered. This state field contains a string of up to 14 characters.

plexes — Each volume has between zero and eight associated plexes.

read policy — A configurable policy for switching between plexes for volume reads. When a volume has more than one enabled associated plex, the Volume

Manager can distribute reads between the plexes to distribute the I / O load and thus increase total possible bandwidth of reads through the volume.

The read policy can be set by administrator. Possible policies are:

round-robin
: For every other read operation, switch to a different plex from the previous read operation. Given three plexes, this will switch between each of the three plexes, in order.

preferred plex
: This read policy specifies a particular named plex that is used to satisfy read requests. In the event that a read request cannot be satisfied by the preferred plex, this policy changes to round-robin.

select
: This read policy is the default policy, and adjusts to use an appropriate read policy based on the set of plexes associated with the volume. If exactly one enabled read-write striped plex is associated with the volume, then that plex is chosen automatically as the preferred plex; otherwise, the round-robin policy is used. If a volume has one striped plex and one non-striped plex, preferring the striped plex often yields better throughput.

start options
: This is a string that is organized as a set of usage-type options to apply when starting (enabling) a volume. See **vxvol**(1M) for details.

log type
: A policy to use for logging changes to the volume, which can be assigned by the administrator. Policies that can be specified are:

  none
  : Do not perform any special actions when writing to the volume. Just write the requested data to all read-write or write-only plexes.

  dirty-region-log
  : A volume is divided into regions. A bitmap where each bit corresponds to a region is maintained. When a write to a particular region occures, the respective bit is set on. When the system is restarted after a crash, this region bitmap is used to limit the amount of data copying that is required to recover plex consistency for the volume. The region changes are logged to special logging subdisks associated with each of the plexes associated with the volume. Use of dirty-region-logging can greatly speed recovery of a volume, but it also degrades performance of the volume under normal operation.

read / write-back recover mode
: This is a mode that applies to the volume, which is managed by utilities as part of plex consistency recovery. When this mode is enabled, each read operation will recover plex consistency for the region covered by the read. Plex consistency is recovered by reading data from blocks of

one plex and writing that data to all other writable plexes. This ensures that a future read operation covering the same range of blocks will read the same data.

write-back-on-read-failure mode

This is a mode that applies to the volume, which can be enabled or disabled by the administrator using **vxedit**. If this mode is enabled, then a read failure for a plex will cause data to be read from an alternate plex and then written back to the plex that got the read failure. This will usually fix the error. Only if the writeback fails will the plex be detached for having an unrecoverable I/O failure.

writecopy mode

This is a mode that applies to the volume, which can be enabled or disabled by the administrator using **vxedit**. This mode takes affect only if dirty-region-logging is in effect. When the operating system hands off a write request to the volume driver, the operating system may continue to change the memory that is being written to disk. The Volume Manager cannot detect that the memory is changing, so it can inadvertently leave plexes with inconsistent contents. This is not normally a problem, because the operating system ensures that any such modified memory is rewritten to the volume before the volume is closed (such as by a clean system shutdown). However, if the system crashes, plexes may be inconsistent. Since the dirty-region-logging feature prevents recovery of the entire volume, it may not ensure that plexes are entirely consistent.

Turning on the writecopy mode (which is normally set by default) often causes the Volume Manager to copy the data for a write request to a new section of memory before writing it to disk. Because the write is done from the copied memory, it cannot change and so the data written to each plex is guaranteed to be the same if the write completes.

exception policy

There are several modes that can be set on the volume, by utilities according to the usage type of the volume. These modes affect operation of a volume in the presence of I/O failures. Currently only one of these policies, called **GEN_DET_SPARSE** is ever used. This policy tracks complete and incomplete plexes in a volume (an incomplete plex does not have a backing subdisk for all blocks in the volume). If an unrecoverable error occurs on an incomplete plex, the plex is detached (disabled from receiving regular volume I/O requests). If an unrecoverable error occurs on a complete plex, the plex is detached unless it is the last complete plex. If the plex is the last complete read-write plex, any incomplete plexes that overlap with the error will be detached but the plex with the error will remain attached.

This default policy is chosen to ensure that an I/O that fails on one plex will not, in the future, be directed to that plex again unless that plex is

the last complete plex remaining attached to the volume. In that case, the policy ensures that the volume will return the error consistently, even in the presence of incomplete plexes.

comment — An administrator-assigned string of up to 40 characters that can be set and changed using the **vxedit** utility. The Volume Manager does not interpret the comment field. The comment cannot contain newline characters.

user, group, and mode — These attributes are the user group and file permission modes used for the volume device nodes, and for the plex device nodes of associated plexes. The user and group are normally **root**. The mode usually allows read and write permission to the owner, and no access by other users.

Plex records

Plex records define the characteristics of a particular mirror of a volume. A plex can be in either an associated state or a dissociated state. In the dissociated state, the plex is not a part of a volume. A dissociated plex cannot be accessed in any way. An associated plex can be accessed through the volume and, in a limited fashion, through a plex device.

Plexes have the following fundamental attributes:

plex state — Each plex is either enabled, disabled, or detached. When enabled, normal read and write operations from the volume can be directed to the plex. When disabled, no I/O operations will be applied to the plex. When detached, normal volume I/O will not be directed to the plex. When detached, the plex device can be accessed for either read or write access using the special plex device nodes. If a plex is enabled, however, then the plex device can be read but not written.

I/O failures encountered during normal volume I/O may move the enabled state for a plex directly from enabled to detached. See the description of volume exception policies for more information.

I/O mode — Each plex is either in read-write, read-only, or write-only mode. This mode affects read and write operations directed to the volume, if the plex is enabled. For read-write and read-only modes, volume read operations can be directed to the plex. For read-write and write-only modes, volume write operations will be directed to the plex.

Plexes are normally in read-write mode. Write-only mode is used to recover a plex that failed, and whose contents have thus become out-of-date with respect to the volume. It is also used when attaching a new plex to a volume. In read-write mode, writes to the volume will update the plex, causing written regions to be up-to-date. Typically, a set of special copy operations will be used to update the remainder of the plex.

layout — The organization of associated subdisks with respect to the plex address space. The layout is either striped or concatenated.

subdisks — Each plex has zero or more associated subdisks. Subdisks are associated at offsets relative to the beginning of the plex address space. Subdisks

for concatenated plexes may not cover the entire length of the plex, leaving holes in the plex. A plex that is not as long as the volume to which it is associated is considered to have a hole at the end of the plex, up to the length of the volume. A plex with a hole is considered incomplete, also sometimes called *sparse*.

log subdisk
: Each plex can have at most one associated log subdisk. A log subdisk is used with the dirty-region-logging feature to improve the time required to recover consistency of a volume after a system failure.

length
: The length of a plex is the offset of the last subdisk in the plex plus the length of that subdisk. In other words, the length of the plex is defined by the last block in the plex address space that is backed by a subdisk. This value may or may not relate to the length of the volume, depending on whether the plex is completely contiguously allocated.

contiguous length
: The offset of the first block in the plex address space that is *not* backed by a subdisk. If the plex has no holes, the contiguous length matches the plex length. If the contiguous length is equal to or greater than the length of the associated volume, the plex is considered complete, otherwise it is sparse.

usage-type state
: Volume usage types maintain a private state field related to the the operations that have been performed on the plex, or to failure conditions that have been encountered. This state field contains a string of up to 14 characters.

condition flags
: Various condition flags are defined for the plex that define state which is recognized automatically, rather than managed by the volume usage type. Defined flags are:

  NODAREC
  : No physical disk could be found corresponding to the disk ID in the disk media record for one of the subdisks associated with the plex. The plex cannot be used until the condition is fixed or the affected subdisk is dissociated.

  REMOVED
  : One of the disk media records was put into the removed state through explicit administrative action. The plex cannot be used until the disk is replaced or the affected subdisk is dissociated.

  RECOVER
  : A disk for one of the disk media records was replaced or was reattached too late to prevent the plex from becoming out-of-date with respect to the volume. The plex requires complete recovery from another plex in the volume to synchronize the plex with the correct contents of the volume.

  IOFAIL The plex was detached as a result of an I/O failure detected

during normal volume I/O. The plex is out-of-date with respect to the volume, and in need of complete recovery. However, this condition also indicates a likelihood that one of the disks in the system should be replaced.

volatile state — A plex is considered to have ''volatile'' contents if the disk for any of the plex's subdisks is considered to be volatile. The contents of a volatile disk are not presumed to survive a system reboot. The contents of a volatile plex are always considered out-of-date after a recovery and in need of complete recovery from another plex.

comment — An administrator-assigned string of up to 40 characters that can be set and changed using the **vxedit** utility. The Volume Manager does not interpret the comment field. The comment cannot contain newline characters.

Subdisk records

Subdisk records define a region of disk, allocated from a disk's public region. Subdisks have very little state associated with them, other than the configuration state that defines which region of disk the subdisk occupies. Subdisks cannot overlap each other, either in their associations with plexes, or in their arrangement on disk public regions.

Subdisks have the following fundamental attributes:

disk media name — The name of the disk media record that the subdisk is defined on.

disk offset — The offset, from the beginning of the disk's public region, to the start of the subdisk.

plex offset — For associated subdisks, this is the offset (from the beginning of the plex) of the subdisk association. For subdisks associated with striped plexes, the plex offset defines relative ordering of subdisks in the plex, rather than actual offsets within the plex address space.

length — The length of the subdisk.

comment — An administrator-assigned string of up to 40 characters that can be set and changed using the **vxedit** utility. The Volume Manager does not interpret the comment field. The comment cannot contain newline characters.

Disk media records

Disk media records define a specific disk within a disk group. The name of a disk media record is assigned when a disk is first added to a disk group (using the **vxdg adddisk** operation). Disk media records can be assigned to specific physical disks by associating the media record with the current disk access record for the physical disk.

Disk media records have the following fundamental attributes:

disk ID — A 64-byte unique identifier representing the physical disk to which the media record is associated. This can be cleared to indicate that the disk is considered in the *removed* state. A removed disk has no current association with any physical disk.

disk access name
: The disk access name that is currently used to access the physical disk referenced by the disk ID. If the disk ID is defined, but no physical disk with that ID could be found, the disk access name will be clear. A disk where the physical disk could not be found is considered to be in the **NODAREC**, or *inaccessible*, state. A disk can become inaccessible either because the indicated disk is not currently attached to the system, or because I/O failures on the physical disk prevented the Volume Manager from identifying or using the physical disk.

A disk media record that has an active association with a physical disk (both the disk ID and the disk access name attributes are defined), inherits several properties from the underlying physical disk. These attributes are taken from the disk header, which is stored in the private region of the the disk. These inherited attributes are:

public length
: The length of the region of the physical disk that is available for subdisk allocations.

private length
: The length of the region of the physical disk that is reserved for storing private Volume Manager information.

atomic I/O size
: This is the fundamental I/O size for the disk, in bytes, also known as the *sector size*. All I/Os destined for this disk must be multiples of this size. Currently, the Volume Manager requires that all disks have the same sector size. On most systems, this size is 512 bytes.

Disk access records

Disk access records define an address, or access path, that can be used to access a disk. The list of all disk access records defines the list of all disk addresses that the Volume Manager can use to locate physical disks. Disk access records do not define specific physical disks, since physical disks can be moved on a system. When a physical disk is moved, a different disk access record may be necessary to locate it.

Disk access records are stored in the **rootdg** disk group configuration. Unlike all other record types, the names of disk access records can conflict with the names of other records. For example, a specialty disk (such as a RAM disk) can use the same name for both the disk access record and the disk media record that points to it. It is typically advisable to use different names for the access and media records, to avoid additional confusion if disks are moved.

Disk access records can be defined explicitly. Some (sometimes all) disk access records may be configured automatically by the Volume Manager, based on available information in the operating system. Such automatically-configured disks are not stored persistently in the on-disk root disk group configuration, but are instead regenerated every time the Volume Manager starts up.

Disk access records have the following fundamental attributes:

disk access name
: The name of the disk access record is typically a disk address of some kind. On Sun systems, disk names are usually of the form **c***C***t***T***d0***s2***, indicating use of the entire disk on controller *C*, with SCSI target ID *T*,

logical unit *D*. The **s2** suffix is used as a convention indicating the use of standard partitioning. Other systems are likely to have different conventions for the disk access name.

type
: Each disk access record has a type, which identifies certain key characteristics of the Volume Manager's interaction with the disk. Currently available types are: **sliced**, **simple**, and **nopriv**. See **vxdisk**(1M) for more information on disk types. Typically, most or all of the disks will be of type **sliced**. It may be desirable to create specialty disks (such as RAM disks) with type **nopriv**.

If the physical disk represented by the disk access record is currently associated with a disk media record, then the following fields are defined:

disk group name
: The name of the disk group containing the disk media record.

disk media name
: The name of the media record that points to the physical disk.

Additional attributes can be added, arbitrarily, by disk types. See **vxdisk** for a list of additional attributes defined by the standard disk types.

VOLUME USAGE TYPES

The usage type of a volume represents a class of rules for operating on a volume. Each usage type is defined by a set of executables under the directory **/etc/vx/type/***usage_type*, where *usage_type* is the name given to the usage type. The required executables are: **vxinfo**, **vxmake**, **vxmend**, **vxplex**, **vxsd**, and **vxvol**. These executables are invoked by the Volume Manager administrative utilities with the same names. The executables under **/etc/vx/type** should not, normally, be executed directly.

Five usage types are provided with the Volume Manager: **gen**, **fsgen**, **root**, **swap**, and **raid5**. It is likely that new usage types will be added in future releases. It is also possible for third-party products to install usage types.

The usage types currently provided with the Volume Manager store state information in the volume and plex usage-type state fields. The state fields defined for volumes are:

EMPTY
: The volume is not yet initialized. This is the initial state for volumes created by **vxmake**.

CLEAN
: The volume has been stopped and the contents for all plexes are consistent.

ACTIVE
: The volume has been started and is running normally, or was running normally when the system was stopped. If the system crashes in this state, then the volume may require plex consistency recovery.

NEEDSYNC
: The volume requires recovery. This is typically set after a system failure to indicate that the plexes in the volume may be inconsistent, so that they require recovery [see the **resync** operation in **vxvol**(1M)].

SYNC
: Plex consistency recovery is currently being done on the volume. **vxvol resync** sets this state when it starts to recovery plex consistency on a volume that was in the **NEEDSYNC** state.

The state fields defined for plexes are:

EMPTY The plex is not yet initialized. This state is set when the volume state is also **EMPTY**.

CLEAN The plex was running normally when the volume was stopped. The plex will be enabled without requiring recovery when the volume is started.

ACTIVE The plex is running normally on a started volume. The plex condition flags (**NODAREC**, **REMOVED**, **RECOVER**, and **IOFAIL**) may apply if the system is rebooted and the volume restarted.

STALE The plex was detached, either by **vxplex det** or by an I/O failure. **vxvol start** will change the state for a plex to **STALE** if any of the plex condition flags are set. **STALE** plexes will be reattached automatically, when starting a volume, by calling **vxplex att**.

OFFLINE The plex was disabled by the **vxmend off** operation. See **vxmend**(1M) for more information.

SNAPATT This is a snapshot plex that is being attached by the **vxassist snapstart** operation. When the attach is complete, the state for the plex will be changed to **SNAPDONE**. If the system fails before the attach completes, the plex and all of its subdisks will be removed.

SNAPDONE This is a snapshot plex created by **vxassist snapstart** that is fully attached. A Plex in this state can be turned into a snapshot volume with **vxassist snapshot**. See **vxassist**(1M) for more information. If the system fails before the attach completes, the plex and all of its subdisks will be removed.

SNAPTMP This is a snapshot plex being attached by the **vxplex snapstart** operation. When the attach is complete, the state for the plex will be changed to **SNAPDIS**. If the system fails before the attach completes, the plex will be dissociated from the volume.

SNAPDIS This is a snapshot plex created by **vxplex snapstart** that is fully attached. A Plex in this state can be turned into a snapshot volume with **vxplex snapshot**. See **vxplex**(1M) for more information. If the system fails before the attach completes, the plex will be dissociated from the volume.

TEMP This is a plex that is being associated and attached to a volume with **vxplex att**. If the system fails before the attach completes the plex will be dissociated from the volume.

TEMPRM This is a plex that is being associated and attached to a volume with **vxplex att**. If the system fails before the attach completes the plex will be dissociated from the volume and removed. Any subdisks in the plex will be kept.

TEMPRMSD This is a plex that is being associated and attached to a volume with **vxplex att**. If the system fails before the attach completes, the plex and

its subdisks will be dissociated from the volume and removed.

EXIT CODES

The majority of the Volume Manager utilities use a common set of exit codes, which can be used by shell scripts or other types of programs to react to specific problems detected by the utilities. For C programmers, these exit status codes are defined in the include file **volclient.h**. The number and macro name for each distinct exit code is described below. Shell script writers must directly compare against the numbers specified.

(0) `VEX_OK` The utility is not reporting any error through the exit code.

(1) `VEX_USAGE` Some command line arguments to the utility were invalid.

(2) `VEX_SYNTAX`
A syntax error occurred in a command or description, or a specified record name is too long or contains invalid characters. This code is returned only by utilities that implement a command or description language. This code may also be returned for errors in search patterns.

(3) `VEX_NOVOLD`
The volume daemon does not appear to be running.

(4) `VEX_IPC` An unexpected error was encountered while communicating with the volume daemon.

(5) `VEX_OSERR` An unexpected error was returned by a system call or by the C library. This can also indicate that the utility ran out of memory.

(6) `VEX_LOST` The status for a commit was lost because the volume daemon was killed and restarted during the commit of a transaction, but after restart the volume daemon did not know whether the commit succeeded or failed.

(7) `VEX_UTILERR`
The utility encountered an error that it should not have encountered. This generally implies a condition that the utility should have tested for but did not, or a condition that results from the volume daemon returning a value that did not make sense.

(8) `VEX_TIMEOUT`
The time required to complete a transaction exceeded 60 seconds, causing the transaction locks to be lost. As most utilities will reattempt the transaction at least once if a timeout occurs, this usually implies that a transaction timed out two or more times.

(9) `VEX_NODG` No disk group could be identified for an operation. This results either from naming a disk group that does not exist, or from supplying names on a command line that are in different disk groups or in multiple disk groups.

(10) `VEX_CHANGED`
A change made to the database by another process caused the utility to stop. This code is also returned by a usage-type-dependent utility if it is given a record that is associated with a different usage-type. If this situation occurs when the usage-type-dependent utility is called from a

switchout utility, then the database was changed after the switchout utility determined the proper usage-type to invoke.

(11) VEX_NOENT
A requested subdisk, plex, or volume record was not found in the configuration database. This may also mean that a record was an inappropriate type.

(12) VEX_EXIST
A name used to create a new configuration record matches the name of an existing record.

(13) VEX_BUSY A subdisk, plex, or volume is locked against concurrent access. This code is used for inter-transaction locks associated with usage-type utilities. The code is also used for the dissociated plex or subdisk lock convention, which writes a non-blank string to the **tutil[0]** field in a plex or subdisk structure to indicate that the record is being used.

(14) VEX_NOUSETYPE
No usage-type could be determined for a utility that requires a usage type.

(15) VEX_BADUSETYPE
An unknown or invalid usage-type was specified.

(16) VEX_ASSOC
A plex or subdisk is associated, but the operation requires a dissociated record.

(17) VEX_DISASSOC
A plex or subdisk is dissociated, but the operation requires an associated record. This code can also be used to indicate that a subdisk or plex is not associated with a specific plex or volume.

(18) VEX_LAST A plex or subdisk was not dissociated because it was the last record associated with a volume or plex.

(19) VEX_TOOMANY
Association of a plex or subdisk would surpass the maximum number that can be associated to a volume or plex.

(20) VEX_INVAL
A specified operation is invalid within the parameters specified. For example, this code is returned when an attempt is made to split a subdisk on a striped plex, or to use a split size that is greater than the size of the plex.

(21) VEX_IOERR
An I/O error was encountered that caused the utility to abort an operation.

(22) VEX_NOPLEX
A volume involved in an operation did not have any associated plexes, although at least one was required.

(23) VEX_NOSUBDISK
A plex involved in an operation did not have any associated subdisks, although at least one was required.

(24) VEX_UNSTARTABLE
A volume could not be started by the **vxvol start** operation, because the configuration of the volume and its plexes prevented the operation.

(25) VEX_STARTED
A specified volume was already started.

(26) VEX_UNSTARTED
A specified volume was not started. For example, this code is returned by the **vxvol stop** operation if the operation is given a volume that is not started.

(27) VEX_DETACHED
A volume or plex involved in an operation is in the detached state, thus preventing a successful operation.

(28) VEX_DISABLED
A volume or plex involved in an operation is in the disabled state, thus preventing a successful operation.

(29) VEX_ENABLED
A volume or plex involved in an operation is in the enabled state, thus preventing a successful operation.

(30) VEX_UNKNOWN
An unknown error was encountered. This code may be used, for example, when the volume daemon returns an unrecognized error number.

(31) VEX_OPEN An operation failed because a volume or plex device was open or mounted, or because a subdisk was associated with an open or mounted volume or plex.

Exit codes greater than 32 are reserved for use by usage-types. Codes greater than 64 can be reserved for use by specific utilities.

SEE ALSO

vxassist(1M), **vxconfigd**(1M), **vxdctl**(1M), **vxdg**(1M), **vxdisk**(1M), **vxdiskadm**(1M), **vxedit**(1M), **vxinfo**(1M), **vxiod**(1M), **vxmake**(1M), **vxmend**(1M), **vxnotify**(1M), **vxplex**(1M), **vxprint**(1M), **vxrecover**(1M), **vxsd**(1M), **vxstat**(1M), **vxtrace**(1M), **vxvol**(1M), **vxwatch**(1M), **plexrec**(4), **sdrec**(4), **vxrec**(4), **vxmake**(4).

For information about other aspects of the Volume Manager see **vxintro**(3X), **vxio**(7), **vxinfo**(7).

NAME

vxiod – start, stop, and report on Volume Manager kernel daemons

SYNOPSIS

vxiod
vxiod [**–f**] **set** *count*

DESCRIPTION

The **vxiod** utility starts, stops, or reports on Volume Manager I/O daemons. An I/O daemon is a process that provides a process context for processing any work that needs to be done to process Volume Manager I/O. Volume Manager I/O daemons are not required for proper operation, though not having any I/O daemons may result in a substantial reduction of system performance.

OPERATION

When invoked with no arguments, **vxiod** prints the current number of volume I/O daemons on the standard output.

When invoked with the **set** keyword, the number of daemons specified by *count* will be created. If more volume I/O daemons exist than are specified by *count*, then the excess processes will be terminated. If more than the maximum number are created (currently 64), the requested number will be silently truncated to that maximum.

The number of daemons to create for general I/O handling is dependent on system load and usage. It is generally not necessary to start more than one daemon for each CPU on the system. If volume recovery seems to proceed slower at some times, then it may be worthwhile to create more daemons.

Each I/O daemon starts in the background and creates an asynchronously-running process, which detaches itself from the controlling terminal and becomes a volume I/O daemon. The **vxiod** utility does not wait for these processes to complete.

OPTIONS

The following options are used by **vxiod**:

-f Force the kill of the last I/O daemon. Without this option, the I/O daemons can only be reduced to one.

EXIT CODES

The **vxiod** utility prints a diagnostic on the standard error, and exits if an error is encountered. If an I/O occurs within a forked I/O daemon process, then the I/O is not reflected in the exit status for **vxiod.** Otherwise, **vxiod** returns a nonzero exit status on errors.

Usage errors result in an exit status of 1 and a usage message. If the requested number of daemons cannot be created, then the exit status is 2, and the number of daemons that were successfully started is reported. If any other error occurs, the exit status is 3.

FILES

/dev/vxiod The device used to report on and start volume I/O daemon processes.

NOTE

Volume Manager I/O daemons cannot be killed directly through the use of signals.

The number of Volume Manager I/O daemons currently running can only be determined by running **vxiod**; I/O daemons to not appear in the list of processes produced by the **ps**(1) command.

SEE ALSO **fork**(2), **vxintro**(1M), **vxio**(7), **vxconfigd**(1M), **vxdctl**(1M), **vxiod**(7).

NAME

vxmake – create Volume Manager configuration records

SYNOPSIS

vxmake [**–Vp**] [**–g** *diskgroup*] [**–o** *useopt*] [**–U** *usetype*] [**–d** *descfile*]
vxmake [**–Vp**] [**–g** *diskgroup*] [**–o** *useopt*] **sd** *name* [*attr*=*value* ...]
vxmake [**–Vp**] [**–g** *diskgroup*] [**–o** *useopt*] **plex** *name* [*attr*=*value* ...]
vxmake [**–Vp**] [**–g** *diskgroup*] [**–o** *useopt*] [**–U** *usetype*] **vol** *name* [*attr*=*value* ...]

DESCRIPTION

The **vxmake** utility creates subdisk, plex, and volume records for the Volume Manager. Records can be created entirely from parameters specified on the command line, or they can be created using a description file.

If no operands are specified, then a description file is used to specify what records to create. By default, this description file is read from the standard input. If operands are specified, then the first operand is a keyword that determines the object to build, the second operand is the name of the object to be created, and additional operands specify attributes for the object to be built. An attribute given on the command line is specified in the same manner used with the description file (with multiple attributes given, one attribute per operand), except that the double-quote convention is not needed. See the ‘‘Examples’’ section for sample uses.

For convenience, a special attribute form can be used for subdisks specified directly from the command line. This special attribute form is:

*/devicepath***,***offset*[**,***len*]

or

*medianame***,***offset*[**,***len*]

The first form specifies a subdisk location by device path, offset within that device, and length. The second form specifies the subdisk location by a disk media record name, offset within the disk’s public region, and length. Optionally, the length can be specified with a **len=***length* attribute instead. If a device path is specified, then it must match the device path for the public region of a known disk in the indicated disk group.

More than one record, including more than one volume or plex hierarchy, can be specified in a single description file. **vxmake** attempts to create all records within a single transaction, so that either all records are created or no records are created.

See the man page **vxmake**(4) for a specification of **vxmake** descriptions.

Descriptions of records within a volume hierarchy are filtered through a usage type specific utility, as appropriate for the usage type, before the records are actually created. Subdisks and plexes that are not specified within a volume hierarchy are filtered by the **gen** usage type. These usage-type utilities may change the descriptions so that certain fields are ignored or set up with initial values. Plex or subdisk records to be associated with a volume or plex may also be changed if the association is indicated in the description, even if the plex or subdisk record is not explicitly specified.

A usage type must be specified for every volume record, either through setting the **usetype** field in the volume record description, or through the **–U** option to **vxmake**.

OPTIONS

The following options are recognized:

–g *diskgroup*
Specify the disk group for the operation. The disk group can be specified either by name or by disk group ID. See **vxdg**(1M) for more information on disk groups.

–U *usetype*
Use *usetype* as the default usage type for volumes. This option has no effect if only subdisks and plexes are created, or if the usage type for a volume is specified directly with the description file variable **usetype**.

–o *useopt*
Give usage-type-specific options to the usage type utility.

–d *desc* Specify the name of a description file to use for building subdisks, plexes, and volumes. If the *desc* option argument is **–d**, then the description file is read from the standard input. If no operands are specified, and no *desc* is specified, then a description file is read from the standard input. This option is ignored for records that are specified directly on the command line.

–V Write a list of utilities that would be called from **vxmake**, along with the arguments that would be passed. The **-V** option performs a ''mock run'' of the utilities; the utilities are not actually called, and no changes are made to the volume configuration database.

–p Print a list of all changes that would be made to the standard output, but do not make those changes. The output is in the **vxmake** description file format. All computable and potentially useful attributes are printed for new records. For records that would be modified, only those attributes that would change are printed, along with an extra attribute declaration, **CHANGED=yes**. In this way the **-p** option performs a ''mock run'' of the utilities.

This option is intended for use with higher-level tools that preview record creations, particularly with respect to usage-type-dependent attribute modifications.

ATTRIBUTE SPECIFICATION RULES

The only attributes that must be specified are the **path** and **len** attributes for subdisk records, and a usage type for volume records. In addition, if the **layout** attribute for a plex record is set to **STRIPE**, then a positive value must be specified for the **stwidth** attribute. Also, if no subdisks are specified for a plex of the **STRIPE** layout, the **ncolumn** attribute must be specified.

Attempts to specify certain attributes are ignored by **vxmake**. Some attributes are ignored by the switchout (usage type independent) **vxmake** itself. The usage type dependent **vxmake** may cause additional attributes to be ignored.

Subdisks

Fields that can be specified for subdisks are: **tutil0**, **tutil1**, **tutil2**, **putil0**, **putil1**, **putil2**, **daname**, **dmname** (or **disk**), **path**, **comment**, **devoffset**, **dmoffset**, **len** and **dmrid**.

The disk specification for subdisks uses one of following fields, in order of precedence: **dmrid**, **dmname**, **daname**, or **path**. The **dmrid** field is used, if non-zero. One of the other fields is used if defined to a non-empty string. If the **path** field is the only field used, then

it must define the block device path for a partition containing the public region of a disk in the requested disk group.

With **path**, the **devoffset** field is used as an offset into the partition; otherwise **dmoffset** is used as an offset into the private region. This distinction is important if the public region does not begin at the beginning of its partition.

Specification of a disk in some form (including possibly the special forms *devicepath*,*offset*,*length* or *medianame*,*offset*,*length*) is required. Specification of the **len** field is also required. All other fields default to zero (for numbers) or empty strings (for strings).

Plexes

Fields that can be specified for plexes are: **tutil0**, **tutil1**, **tutil2**, **putil0**, **putil1**, **putil2**, **comment**, **layout**, **ncolumn**, **stwidth**, **sd**, and **logsd**.

There are no required fields for plex records. All fields default to zero (for numbers) or empty strings (for strings). The **layout** field defaults to **concat**.

If the **layout** field is set to **stripe** or **raid5**, then the **stwidth** attribute must be specified. If the **layout** field is set to **concat**, the **ncolumn** field cannot be specified.

An **sd** specification names the subdisks to associate with the plex. The specification optionally names the offsets of the subdisks within the plex. If no offsets are specified, then the subdisks are directly concatenated in the order indicated. If the **sd** attribute is specified for a striped plex and **ncolumn** is not specified, **ncolumn** will default to the column of the subdisk with the highest column specification. As a special case, if **ncolumn** is not specified and the **sd** does not specify columns or offsets for the subdisks, then each subdisk will be placed in it's own column at an offset of zero in the order they are specified and the plex's **ncolumn** attribute is set to the number of subdisks specified.

A **logsd** specification names a subdisk to associate as a special logging subdisk for recording volume activity as part of the dirty region logging feature.

Volumes

Fields that can be specified for volumes are: **tutil0**, **tutil1**, **tutil2**, **putil0**, **putil1**, **putil2**, **comment**, **usetype**, **startopts**, **readpol**, **prefname**, **minor**, **user**, **group**, **mode**, **len**, **writeback**, **writecopy**, **logtype**, and **plex**.

Specification of a usage type is required, either using the **–U** option on the command line, or using the **usetype** attribute. Specification of a **usetype** attribute overrides a value specified with **–U**. All other fields have default values: **readpol** defaults to **round**; **user** and **group** both default to 0 (**root**); **mode** defaults to **u=rw** (mode 600); **writeback** and **writecopy** default to **on**; the **len** field defaults to the length of the shortest associated plex (or zero, if no plex associations are requested); all other numeric fields default to zero, and all other string fields default to empty strings.

A **plex** specification names the plexes to associate with the volume. The order of plexes in this list is not important.

The value of the **prefname** field is relevant only if the **readpol** attribute is set to **prefer**.

EXAMPLES

The following is an example of a **vxmake** description file:

```
#rectyp  #name     #options
sd       disk3-01  disk=disk3  offset=0  len=10000
sd       disk3-02  disk=disk3  offset=25000  len=10480
sd       disk4-01  disk=disk4  offset=0  len=8000
sd       disk4-02  disk=disk4  offset=15000  len=8000
sd       disk4-03  disk=disk4  offset=30000  len=4480
plex     db-01     layout=STRIPE  ncolumn=2  stwidth=16k
                   sd=disk3-01:0/0,disk3-02:0/10000,disk4-01:1/0,disk4-02:1/8000,disk4-03:1/1
sd       ramd1-01  disk=ramd1  len=640
                   comment="Hot spot for dbvol
plex     db-02     sd=ramd1-01:40320

vol      db        usetype=gen  plex=db-01,db-02
                   readpol=prefer  prefname=db-02
                   comment="Uses mem1 for hot spot in last 5m
```

This description specifies a *gen* type volume that contains two plexes: a volatile memory disk plex, which is preferred, and a physical disk plex. The memory disk plex is sparse and covers only the last 640 sectors of the 40960-sector length of the volume. The physical disk plex is striped across two 20480-sector columns. Column zero contains two subdisks and column one contains three subdisks. The plex offset specified for each subdisk concatenates the subdisks contiguously in their respective columns.

For striped or raid-5 plex subdisk associations if one number is specified for the the column and column offset field, the number is interpreted as a column number and the subdisk is associated at the end of the column in the order it appears in the subdisk list. The same subdisk association list for plex **db-01** above could be re-written as follows:

sd=disk3-01:0,disk3-02:0,disk4-01:1,disk4-02:1,disk4-03:1

This is a simple sequence of commands to create a hierarchy with one subdisk, one plex, and one volume:

```
volmake sd 5s1-01 /dev/dsk/5s1,32000 comment="5s1 subdisk 1"
volmake plex demo-1 sd=5s1-01 comment="Demo volume, plex 1"
volmake –U gen vol demo plex=demo-1 comment="Demo gen volume"
```

FSGEN AND GEN DESCRIPTION

The **fsgen** and **gen** usage-type-specific utilities that support **vxmake** are used by the switchout **vxmake** utility as filters that set defaults for plex and volume states.

Subdisk fields are set, checked, or modified as follows:

len Subdisk lengths are required to be set to a nonzero value.

tutil0 This field is cleared.

Plex fields are set, checked, or modified as follows:

tutil0 This field is cleared.

state If the plex is to be associated with a volume, the **state** field is set to **EMPTY**; otherwise it is cleared.

Volume fields are set, checked, or modified as follows:

tutil0 This field is cleared.

state This field is set to **EMPTY**.

Attempts to associate an existing subdisk or plex with a new plex or volume will be refused, unless the **putil0** field for the existing record is empty.

The **putil0** fields are not set, and their contents are preserved.

FILES

/etc/vx/type/*usetype***/vxmake**
Usage-type-specific utility for filtering volume hierarchy descriptions.

RAID5 DESCRIPTION

the **raid5** usage-type-specific utilities that support **vxmake** are used by the switchout **vxmake** utility as filters that set defaults for plex and volume states.

Subdisk fields are set, checked or modified as follows:

len Subdisk lengths are required to be set to a nonzero value.

tutil0 This field is cleared.

Plex fields are set, checked, or modified as follows:

tutil0 This field is cleared.

state If the plex is to be associated with a volume, the **state** field is set to **EMPTY**; otherwise it is cleared.

Volume fields are set, checked, or modified as follows:

tutil0 This field is cleared.

state This field is set to **EMPTY**.

Attempts to associate an existing subdisk or plex with a new plex or volume will be refused, unless the **putil0** field for the existing record is empty.

The **putil0** fields are not set, and their contents are preserved.

The **raid5** specific **vxmake** utility will only allow one plex with a layout of **raid** to be associated with a volume. Plexes not of the **raid** layout are associated as Raid-5 logging plexes.

EXIT CODES

The utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility.

See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

vxintro(1M), **vxassist**(1M), **vxdisk**(1M), **vxdg**(1M), **vxmake**(4).

NAME

vxmend – mend simple problems in configuration records

SYNOPSIS

vxmend [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–Vfrvps**] **clear** *field name ...*
vxmend [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–Vfrvp**] **fix** *how name* [*arg ...*]
vxmend [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–Vfrvp**] **off** *name ...*
vxmend [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–Vfrvp**] **on** *name ...*

DESCRIPTION

The **vxmend** utility performs various Volume Manager usage-type-specific operations on subdisk, plex, and volume records. The first operand is a keyword that determines the specific operation to perform. The remaining operands specify the configuration objects to which the operation is applied.

Each invocation can be applied to only one disk group at a time, due to internal implementation constraints. Any *name* operands will be used as record names to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). A specific disk group can be forced with **–g** *diskgroup*.

These are the recognized operation keywords:

clear Clear specified utility fields for each named record in the disk group. An option of **–v**, **–p**, or **–s** specifies that the utility operates only on volumes, plexes, or subdisks, respectively. If a record is a volume, or is associated directly or indirectly with a volume, then the **clear** operation is performed according to rules used by the usage type appropriate for that volume.

The *field* operand is a comma-separated list of keywords specifying fields to be cleared. Each keyword in the *field* operand is one of the following:

all Clear all clearable utility fields. Normally, this clears all of the persistent and non-persistent utility fields. A usage-type utility may choose a different set of fields, as appropriate.

putil Clear all persistent utility fields.

tutil Clear all non-persistent (temporary) utility fields.

putil0, **putil1**, or **putil2**
Clear a specific persistent utility field.

tutil0, **tutil1**, or **tutil2**
Clear a specific non-persistent utility field.

Usage types may implement additional field keywords. A usage type may also limit the set of clear operations that can be performed.

fix Change the state of a volume or plex, named by the *name* operand, in a manner specified by the *how* string. The meaning of this operation is entirely usage-type specific. This operation applies only to volumes, or to plexes associated with a volume. Usage type rules appropriate for the volume are used to interpret the command. Additional arguments, after *name*, are interpreted according to rules defined by the usage type.

off Put the named plexes or volumes into an offline state. This operation can be

applied only to volumes, or to plexes associated with a volume. Usage type rules appropriate for the volume are used to perform the operation.

on Take the named plexes or volumes out of the offline state. This operation can be applied only to volumes, or to plexes associated with a volume. Usage type rules appropriate for the volume are used to perform the operation.

OPTIONS

The following options are recognized:

–g *diskgroup* Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *name* operands.

–U *usetype* Limit the operation to apply to this usage type. Attempts to affect volumes with a different usage type will fail.

–o *useopt* Pass in usage-type-specific options to the operation.

–V Write a list of utilities that would be called from **vxmend**, along with the arguments that would be passed. The **–V** performs a ''mock run'' so the utilities are not actually called.

–f Force an operation that the Volume Manager considers potentially dangerous or unnecessary. This enables a limited set of additional operations. Some operations may be disallowed even with this flag.

–r Operate recursively on records associated with the named volume or plex record. Operations applied to a volume will apply to the associated plexes and subdisks. Likewise, operations applied to a plex may be applied to the associated subdisks.

–v Require that *name* operands name volume records.

–p Require that *name* operands name plex records.

–s Require that *name* operands name subdisk records.

FSGEN AND GEN USAGE-TYPES

The **fsgen** and **gen** usage types provide identical semantics for all operations of the **vxmend** utility. These usage types provide the following options as arguments to **–o**:

force Force an operation that internal consistency checks consider to be questionable. This applies to attempts to use **vxmend fix empty** to uninitialize a volume that has plexes in the **ACTIVE** state, and also to attempts to disable the last plex, or the last complete (non-sparse) plex, in a volume. This flag is the same as **–f**.

vol=*volume* Require that any named plex or subdisk record be associated, directly or indirectly, with a volume named *volume*. Several **vol** options can be specified to indicate a list of allowed volume names.

plex=*plexname* Require that any named subdisk record be associated with a plex named *plexname*. Several **plex** options can be specified to indicate a list of allowed plex names.

Limitations and extensions for the **fsgen** and **gen** usage types consist of the following:

clear Keywords supported in the *field* operands include all standard keywords. In addition, a keyword of the form **att***number* can be used to remove locks on a volume for a particular number of concurrent plex attach operations. For example, if the command:

vxplex att v1 v1-01

is aborted by pressing the INTERRUPT key several times (which prevents a clean abort of the operation), you may have to clear the operation with:

vxmend clear tutil0 v1-01
vxmend clear att1 v1
vxplex dis v1-01

Interruption of the above-mentioned **vxplex att** command will display a list of commands to run, which will include the previous list.

The number of plexes currently being attached, either directly or as part of a compound operation, is stored in the **tutil0** field of the volume record as a string in the form **ATT***number*. If the number drops to zero, the **tutil0** field is cleared. Some operations on a volume require that the **tutil0** field be cleared.

fix The **fsgen** and **gen** usage types support the following **vxmend fix** operations:

vxmend fix clean *plex*

Set the state for the named plex to **CLEAN**. The associated volume must be disabled, the named plex must be in the **STALE** state, and the volume must have no additional plexes in the **CLEAN** state.

A volume is not startable if one plex is in the **CLEAN** state and some plexes are in the **ACTIVE** state. Thus, several **vxmend fix** operations are normally used in conjunction to set all plexes in a volume to **STALE** and then to set one plex to **CLEAN**. A **volume start** operation will then enable the **CLEAN** plex and recover the **STALE** plexes by copying data from the one **CLEAN** plex.

vxmend fix active *plex*

Set the state for the named plex to **ACTIVE**. The state for the volume is set to **SYNC**. The associated volume must be disabled, and the named plex must be in the **STALE** state.

When starting a volume in the **SYNC** state, all **ACTIVE** plexes are enabled and are synchronized to have the same contents using a special read / write-back recovery mode. Any **STALE** plexes are then recovered by copying data from the **ACTIVE** plexes.

vxmend fix stale *plex*

Set the state for the named plex to **STALE**. The associated volume must be disabled, and the named plex must be **ACTIVE** or **CLEAN**. This operation names plexes that will be recovered by copying data from other plexes by a **vxvol start** operation.

vxmend fix empty *volume*
This operation sets the named volume and all of its associated plexes to the **EMPTY** state. The volume can then be re-initialized using **vxvol start** or any of the **vxvol init** operations. This operation requires that the volume be disabled.

off The **fsgen** and **gen** usage types allow volumes and plexes to be specified as operands to **vxmend off**. A volume can be named only if **–r** is used to specify recursion, and is applied to all plexes in the volume. Offlining a plex disables the plex and sets its state to **OFFLINE**.

Offlining the last enabled read-write plex in a volume, or the last complete plex (if there are additional sparse plexes) will normally fail unless **–f** is specified. Applying this operation to a volume also disables the volume, and does not require use of **–f**.

If a volume is disabled, then the checks for the last plex or the last complete plex are performed on the set of **ACTIVE** and **CLEAN** plexes, rather than on enabled read-write plexes.

An offline state for a plex can be cleared with **vxmend on**. Alternatively, an offline plex can be reattached with **vxplex att**.

on The **fsgen** and **gen** usage types allow volumes and plexes to be specified as operands to **vxmend on**. The **vxmend on** operation applied to a plex will change the **OFFLINE** state for a plex to **STALE**, allowing the plex to be recovered by the next **vxvol start** or **vxvol startall**. Applying **vxmend on** to a volume will change the state for all associated **OFFLINE** plexes to **STALE**.

RAID5 USAGE-TYPE

The **raid5** usage type provides the following options as arguments to **–o**:

force Force an operation that internal consistency checks consider to be questionable.

In addition to the operations supported for the **fsgen** and **gen** usage types, the **raid5** usage type also supports the following extension:

fix In addition to those supported for the **fsgen** and **gen** usage types, the **raid5** usage type supports the following **vxmend fix** operation:

vxmend fix unstale *subdisk*
Clear any flags indicating that a subdisk of a RAID-5 plex is invalid. This will prevent the data on the subdisk from being recovered when the volume is started.

FILES

/etc/vx/type/*usetype***/vxmend**
The utility that performs **vxmend** operations for a particular volume usage type.

EXIT CODES

The utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered but rather denotes the first condition that prevented further execution of the utility. See **vxintro**(1M) for a list of

standard exit codes.

SEE ALSO

vxintro(1M), **vxplex**(1M), **vxvol**(1M).

NAME

vxmirror – mirror volumes on a disk or control default mirroring

SYNOPSIS

/etc/vx/bin/vxmirror [**–g** *diskgroup*] [**–d yes** | **no**] *medianame* [*new_medianame* ...]
/etc/vx/bin/vxmirror [**–g** *diskgroup*] [**–d yes** | **no**] **-a** [*new_medianame* ...]
/etc/vx/bin/vxmirror [**–g** *diskgroup*] [**–d yes** | **no**]
/etc/vx/bin/vxmirror [**–g** *diskgroup*] **-D**

DESCRIPTION

The **vxmirror** command provides a mechanism to mirror all the contents of a specified disk, to mirror all currently un-mirrored volumes in the specified disk group, or to change or display the current defaults for mirroring. All volumes that have only a single plex (mirror copy), will be mirrored by adding an additional plex.

Volumes containing subdisks that reside on more than one disk will not be mirrored by **vxmirror**.

vxmirror is generally called from the **vxdiskadm** menus. It is not an interactive command and once called, will continue until completion of the operation or until a failure is detected. Due to the nature of generating mirror copies of volumes, this command may take a considerable time to complete.

Mirroring volumes from the boot disk produces a disk that can be used as an alternate boot disk. This is done by calling the **vxrootmir** command. This will mirror root and swap volumes. If usr and var are also volumes they too will be mirrored even though they may not be on the same disk.

In the first listed form of the command, the disk media name is supplied on the command line to **vxmirror**. That name is taken to be the only disk from which volumes should be mirrored. In the case of mirroring volumes from a specified disk, only simple single-subdisk volumes are mirrored.

In the first and second listed forms of the command, the "*new_medianame* ..." parameter identifies a new disk media name or set of names. The mirroring operation being performed will use the names as targets on which to allocate the mirrors. An error will result if the same disk is specified for both the source and target disk and if no other viable targets are supplied.

vxmirror supports the following options:

–g *diskgroup*
: Limit operation of the command to the given disk group, as specified by disk group ID or disk group name. The *medianame* operands will be evaluated relative to the given disk group. If no disk group is supplied to the **vxmirror** command, then **rootdg** is presumed.

–d
: Change the default for subsequent volume creation, depending on the option argument. If the option argument is **yes**, then all subsequent volumes created using the **vxassist** command will automatically be created as mirrored volumes. If the option argument supplied is **no**, then mirroring will be turned off by default.

–a
: Mirror all existing volumes for the specified disk group.

EXAMPLES

The following are examples of the use of the **vxmirror** command:

vxmirror -d yes disk01

This invocation will cause the disk named **disk01** to have its contents mirrored to any available space on any available disk. Subsequent calls to **vxassist** will cause created volumes to be mirrored by default.

vxmirror -D

This command will display the current status of default mirroring. It will output the string *yes*, if mirroring is currently enabled or *no* if not.

vxmirror disk02 disk03

This will cause any volumes on **disk02** to be mirrored to **disk03**.

FILES

/etc/default/vxassist The defaults file for **vxassist** parameters

SEE ALSO

vxintro(1M), **vxassist**(1M), **vxrootmir**(1M).

NAME

vxnotify – display Volume Manager configuration events

SYNOPSIS

vxnotify [**–icfdD**] [**–w** *wait-time*] [**–g** *diskgroup*] [**–n** *number*] [**–t** *timeout*]

DESCRIPTION

The **vxnotify** utility displays events related to disk and configuration changes, as managed by the Volume Manager configuration daemon, **vxconfigd**. **vxnotify** displays requested event types until killed by a signal, until a given number of events have been received, or until a given number of seconds have passed.

Each event is displayed as a single-line output record on the standard output. Displayed events are:

connected

A connection was established with **vxconfigd**. This event type is displayed immediately after successful startup and initialization of **vxnotify**. A **connected** event is also displayed if the connection to **vxconfigd** is lost, and then regained. A **connected** event displayed after a reconnection indicates that some events may have been lost.

disconnected

The connection to **vxconfigd** was lost. This normally results from **vxconfigd** being stopped (such as by **vxdctl stop**) or killed by a signal. In response to a disconnection, **vxnotify** displays a **disconnected** event and then waits until a reconnection succeeds. A **connected** event is then displayed.

A **disconnected** event is also printed if **vxconfigd** is not accessible at the time **vxnotify** is started. In this case the **disconnected** event precedes the first **connected** event.

more events

Due to internal buffer overruns, or other possible problems, some events may have been lost.

vxconfigd disabled

vxconfigd was changed to disabled mode. Most configuration information will be unavailable until **vxconfigd** is changed back to enabled mode.

vxconfigd enabled

vxconfigd was changed to enabled mode. All configuration information should now be retrievable. **vxconfigd disabled** and **vxconfigd enabled** events can be retrieved only when using diagnostic-mode connections to the **vxconfigd** diagnostic portal. Use **–D** to obtain a regular diagnostic mode connection.

waiting ...

If the **–w** option is specified, a **waiting** event is displayed after a defined period with no other events. Shell scripts can use **waiting** messages to collect groups of related, or at least nearly simultaneous, events. This can make shell scripts more efficient. This can also provide some scripts with better input since sets of detach events, in particular, often occur in groups that scripts can relate together. This is particularly important given that a typical shell script will block until **vxnotify**

produces output, thus requiring output to indicate the end of a possible sequence of related events.

import dg *groupname* **dgid** *groupid*
The disk group named *groupname* was imported. The disk group ID of the imported disk group is *groupid*.

deport dg *groupname* **dgid** *groupid*
The named disk group was deported. The disk group ID of the imported disk group is *groupid*.

disable dg *groupname* **dgid** *groupid*
The named disk group was disabled. A disabled disk group cannot be changed, and its records cannot be printed with **vxprint**. However, some volumes in a disabled disk group may still be usable, although it is unlikely that the volumes will be usable after a system reboot. A disk group will be disabled as a result of excessive failures. A disk group will be disabled if the last disk in the disk group fails, or if errors occur when writing to all configuration and log copies in the disk group.

change dg *groupname* **dgid** *groupid*
A change was made to the configuration for the named disk group. The transaction ID for the update was *groupid*.

detach subdisk *subdisk* **plex** *plex* **volume** *volume* **dg** *groupname* **dgid** *groupid*
The named subdisk, in the named disk group, was detached as a result of an I/O failure detected during normal volume I/O, or disabled as a result of a detected disk failure. Failures of a subdisk in a RAID-5 volume or a log subdisk within a mirror volume will result in a subdisk detach; other subdisk failures generally result in the subdisk's plex being detached.

detach plex *plex* **volume** *volume* **dg** *groupname* **dgid** *groupid*
The named plex, in the named disk group, was detached as a result of an I/O failure detected during normal volume I/O, or disabled as a result of a detected total disk failure.

detach volume *volume* **dg** *groupname* **dgid** *groupid*
The named volume, in the named disk group, was detached as a result of an I/O failure detected during normal volume I/O, or as a result of a detected total disk failure. Usually, only plexes or subdisks are detached as a result of volume I/O failure. However, if a volume would become entirely unusable by detaching a plex or subdisk, then the volume may be detached.

detach disk *accesname* **dm** *medianame* **dg** *groupname* **dgid** *groupid*
The named disk, with device access name *accessname* and disk media name *medianame* was disconnected from the named disk group as a result of an apparent total disk failure. Total disk failures are checked for when plexes or subdisks are detached by kernel failures, or, explicitly, by the **voldisk check** operation (see **voldisk**(1M)).

log-detach volume *volume* **dg** *groupname* **dgid** *groupid*
All log copies for the volume (either log plexes for a RAID-5 volume or log subdisk for a regular mirrored volume) have become unusable, either as a result of I/O failures or as a result of a detected total disk failure.

change disk *accessname* **dm** *medianame* **dg** *groupname* **dgid** *groupid*
The disk header changed for the disk with a device access name of *accessname.* The disk group name and ID of the disk are *groupname* and *groupid,* respectively. The displayed *groupname* and *groupid* strings will be ''**-**'' or blank if the disk is not currently in an imported disk group.

degraded volume*volume* **dg** *groupname* **dgid** *groupid*
The RAID-5 volume has become degraded due to the loss of one subdisk in the **raid5** plex of the volume. Accesses to some parts of the volume may be slower than to other parts depending on the location of the failed subdisk and the subsequent I/O patterns.

OPTIONS

The following options are recognized:

–i Display disk group import, deport, and disable events.

–c Display disk group change events.

–f Display plex, volume, and disk detach events.

–d Display disk change events.

–D Use a diagnostic-mode connection to **vxconfigd**. This allows the receipt of events when vxconfigd is running in disabled mode. Access to configuration information is limited when **vxconfigd** is running in disabled mode. For most applications, it is better to let **vxnotify** print events only when **vxconfigd** is running in enabled mode.

–w *wait_time*
Display **waiting** events after *wait_time* seconds with no other events.

–g *diskgroup*
Restrict displayed events to those in the indicated disk group. The disk group can be specified either as a disk group name or a disk group ID.

–n *number*
Display the indicated number of **vxconfigd** events, then exit. Events that are not generated by **vxconfigd** (i.e., connect, disconnect and waiting events) do not count towards the number of counted events and will not cause an exit to occur.

–t *timeout*
Display events for up to *timeout* seconds, then exit. The **–n** and **–t** options can be combined to specify a maximum number of events and a maximum timeout to wait before exiting.

If none of the **–i**, **–c**, **–f**, or **–d** options are specified, then default to printing all event types. If a disk group is specified with **–g**, display only disk group-related events.

EXAMPLES

The following example shell script will send mail to **root** for all detected plex, volume, and disk detaches:

```
checkdetach() {
  d=‘vxprint -AQdF ’%name %nodarec’ | awk ’$2=="on" {print " " $1}’‘
  p=‘vxprint -AQpe ’pl_kdetach | | pl_nodarec’ -F ’ %name’‘
  v=‘vxprint -AQvF ’ %name’ -e \
          "((any aslist.pl_kdetach==true) | |
           (any aslist.pl_nodarec)) &&
           !(any aslist.pl_stale==false)"‘

  if [ ! -z "$d" ] | | [ ! -z "$p" ] | | [ ! -z "$v" ]
  then
    (
      cat <<EOF
Failures have been detected by the VERITAS Volume Manager:
EOF
      [ -z "$d" ] | | echo "\ \nfailed disks:\ \n$d"
      [ -z "$p" ] | | echo "\ \nfailed plexes:\ \n$p"
      [ -z "$v" ] | | echo "\ \nfailed volumes:\ \n$v"
    ) | mailx -s "Volume Manager failures" root
  fi
}

vxnotify -f -w 30 | while read code more
do
  case $code in
  waiting) checkdetach;;
  esac
done
```

EXIT CODES

The **vxnotify** utility exits with a nonzero status if an error is encountered while communicating with **vxconfigd**. See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

vxintro(1M), **vxconfigd**(1M), **vxtrace**(1M), **vx_notify**(3X).

NAME

vxplex – perform Volume Manager operations on plexes

SYNOPSIS

vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **att** *volume plex ...*
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] **det** *plex ...*
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] **dis** *plex ...*
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **cp** *volume plex ...*
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **snapstart** *volume plex*.
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **snapdone** *plex newvolume*
vxplex [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] **mv** *oldplex newplex*

DESCRIPTION

The **vxplex** utility performs Volume Manager operations on plexes and on volume-and-plex combinations. The first operand is a keyword that determines the specific operation to perform. The remaining operands specify the configuration objects to which the operation is to be applied.

Each operation can be applied to only one disk group at a time, due to internal implementation constraints. Any *volume* or *plex* operands will be used to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). A specific disk group can be forced with **–g** *diskgroup*.

These are the recognized operation keywords:

att Attach each named plex to the named volume. This can be applied to dissociated plexes, or to non-enabled plexes already associated with the named volume. The operation can not be applied to a sparse, striped plex. If the volume is enabled, then the result of the successful operation will be to associate the plex (if needed) and to recover the plex to have the same contents as all other attached plexes in the volume. The rules for performing the attach depend upon the usage type of the named volume.

Attaching a plex is the normal means of recovering a plex after a disk replacement, or after a plex offline.

det Detach each of the named plexes. Detaching a plex leaves the plex associated with its volume, but prevents normal volume I/O from being directed to the plex. If the volume is enabled or detached, then the plex will be accessible directly. This operation can be applied to plexes that are enabled or disabled. The rules for performing the detach depend upon the usage types of the volumes involved. The operation does not apply to dissociated plexes.

Detaching a plex can be used as a simple means of backing up a volume that is normally mirrored. With this type of procedure, the detached plex (which represents a consistent image of the volume exactly at the time of the detach) is backed up using the plex device, and is then reattached when the backup operation completes. Some usage types will perform special synchronization operations to help make the detached plex consistent with respect to in-memory data, which can help improve the quality of such backups.

dis Dissociate each of the named plexes. Dissociating a plex breaks the link between the plex and its volume. A dissociated plex is inaccessible until it is reassociated,

which can be done either with **vxplex att** or with **vxmake**. Any checks and synchronizations that apply to the **det** operation also apply to the **dis** operation.

Plex dissociation is the normal means of unmirroring a volume, or reducing the mirror count for a volume. To support this use, **–o rm** can be used to dissociate and remove the plex (and its associated subdisks) in the same operation. This makes the space used by those subdisks usable for new allocations (such as with **vxassist** or with **vxmake**).

Plex dissociation can also be used for file system backups of volumes that are normally mirrored. Plex devices are not directly mountable, so the backup method described for the **det** operation will not work if the backup program requires a mounted file system. To support such backup programs, a plex can be dissociated and can then be allocated to a new volume, such as with the command:

vxmake -U gen vol *volume* **plex=***plex*

The created volume can then be started and mounted for use by the backup program.

cp Copy the named volume to the named plexes. The volume cannot be enabled, and the named plexes must not be associated. The results of the operation will be a set of dissociated plexes that are an exact copy of the volume at the time of completion of the operation. The rules for performing the attach depend upon the usage type of the named volume. To improve the quality of the copies, some usage types attempt to make the detached plex consistent with respect to in-memory data. The operation can not be performed to a sparse, striped plex.

This operation can be used to make a copy of a volume, for backup purposes, without mirroring the volume in advance.

snapstart and **snapdone**

These two operations form the two parts of a preferred means of copying a volume to a plex for backup purposes. The **snapstart** operation attaches a plex to a volume and, when the operation is complete, leaves the plex associated as a temporary plex. After the operation completes, the administrator can convert the plex attached by **snapstart** into a new volume using **vxplex snapdone**. To improve the quality of the copies, some usage types attempt to make the detached plex consistent with respect to in-memory data.

This method of backup is preferable to using **vxplex cp** because it allows the administrator to coordinate breaking off the plex from the original volume at a well-defined point in time. This is important, since attaching a plex to a volume can take a considerable amount of time, and it is difficult to know when it will complete. Also, direct conversion of the plex into a new volume is more convenient than requiring additional steps.

mv Attach the plex *newplex* to the volume that *oldplex* is associated with and dissociate *oldplex*. The volume cannot be disabled, and *newplex* must name a dissociated plex. The operation can not be applied to a sparse, striped plex. The operation ensures seamless replacement of the dissociated plex without loss of data in the

volume and without significant delays in volume accessibility.

A primary purpose for the plex move operation is to move a plex that is using a disk to another location. In support of this purpose for the operation, **–o rm** can be specified to remove the original plex after completion of the operation.

For concatenated or striped plexes the **vxsd mv** operation can be used to move individual subdisks off a disk. The rules for performing the move depend upon the usage types of the volume to which *oldplex* is associated.

OPTIONS

The following options are recognized:

–g *diskgroup* Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *name* operands.

–U *usetype* Limit the operation to apply to this usage type. Attempts to affect volumes with a different usage type will fail.

–o *useopt* Pass in usage-type-specific options to the operation. A certain set of operations are expected to be implemented by all usage types:

slow[**=***iodelay*]
Reduce the system performance impact of copy operations. Copy operations are usually a set of short copy operations on small regions of the volume (normally from 16 kilobytes to 128 kilobytes). This option inserts a delay between the recovery of each such region. A specific delay can be specified with *iodelay* as a number of milliseconds; otherwise, a default is chosen (normally 250 milliseconds).

iosize=*size*
Perform copy operations in regions with the length specified by *size*, which is a standard Volume Manager length number (see **vxintro**(1M)). Specifying a larger number typically causes the operation to complete sooner, but with greater impact on other processes using the volume. The default I/O size is typically 32 kilobytes.

rm Remove the plexes after successful completion of a **vxplex dis** operation. Remove the source plex after successful completion of **vxplex mv**.

–v *volume* Require that the plex named by a *plex* or *oldplex* operand be associated with the named volume. This option can be used as a sanity check, to ensure that the specified plex is actually the plex desired for the operation.

–V Write a list of utilities that would be called from **vxplex**, along with the arguments that would be passed. The **–V** performs a ‘‘mock run’’ so the utilities are not actually called.

–f Force an operation that the Volume Manager considers potentially

dangerous or of questionable use. This permits a limited set of operations that would otherwise be disallowed. Some operations may be disallowed even with this flag.

FSGEN AND GEN USAGE-TYPES

The **fsgen** and **gen** usage types provide similar, though not identical, semantics for all operations of the **vxplex** utility. In particular, the **fsgen** usage type will attempt to flush in-memory data cached for the file system residing on the volume. For most file systems, this consists of calling **sync**(1M) to attempt to flush all in-memory data to disk. For the **vxfs** file system type, this will use special ioctls to ensure a reliable flush of the involved volume.

If a **vxplex** operation is interrupted by a signal, then an attempt is made to restore the disk group configuration to a state that is roughly equivalent to its original state. If this attempt is interrupted (such as through another signal) then the user may need to perform some cleanup. The specific cleanup actions that are needed are written to the standard error before **volplex** exits.

The **fsgen** and **gen** usage types provide the following options as arguments to **–o** in addition to the required options:

force Force an operation that the Volume Manager considered potentially dangerous or of questionable use. This applies to attempts to detach or dissociate the last (complete) plex in a volume, or to attempts to move a plex to a plex that has a different size. This flag is the same as **–f**.

rerr Ignore volume or plex read errors when copying data onto a plex. A warning message is written to standard error if a read error occurs, but the error does not affect success of the operation. This operation can be used only with the **cp** operation; the operation is ignored if used with other operations.

werr Ignore plex write errors when copying data onto a plex. A warning message is written to standard error if a write error occurs, but the error does not affect success of the operation. This operation can be used only with the **cp** operation; the operation is ignored if used with other operations.

mapzero If a plex is moved to a new plex that has regions that are mapped to a subdisk in the destination, but are not mapped to a subdisk for any enabled, readable plex in the volume, then zero out that mapped region in the destination plex. Without this flag, the mapped region may be left unchanged from its original contents.

Limitations and extensions for the **fsgen** and **gen** usage types consist of the following:

att If the volume is enabled and one of the named plexes is associated with the volume, then the plex must be **STALE**, **EMPTY**, **ACTIVE**, or **OFFLINE**. If the operation succeeds in attaching a plex, then any I/O fail condition for the plex is cleared. Also, attaching to an enabled volume requires that the volume have at least one enabled, read-write plex.

If the volume is not enabled, then the named plexes are associated with the

volume (if not already associated) and are set to the **STALE** state, so that the plex will be fully attached by the next **vxvol start** or **vxvol startall** operation that applies to the volume.

If the logging type of the volume is *UNDEF* and an unassociated plex with a logging subdisk is attached, the volume is automatically converted to have a logging type of *DRL*. Logging of volume changes is enabled when the volume and has at least one enabled, associated plex with an enabled log subdisk and at least two read-write mode plexes.

An attempt to attach an unassociated plex fails if the **putil0** field is not empty. This makes it possible to prevent use of a plex by using **vxedit set** to set the **putil0** field to a non-empty string. The **putil0** field can then be cleared with either **vxedit set** or with **vxmend clear putil0**.

dis and **det**

A detach or dissociate of a plex in an enabled volume fails if applied to a plex that is the last complete, enabled, read-write plex in the volume and the volume contains two or more non-complete, enabled, read-write plexes. In other words, a volume cannot be left with two enabled, non-complete plexes. A complete plex is one that is at least as long as the volume, and has subdisks mapped to the plex for all blocks up to the length of the volume. The **–f** option is reuired to reduce a volume to containing one enabled, read-write, non-complete plex, or to having no enabled, read-write plexes at all.

The **det** operation changes the state for an **ACTIVE** or **CLEAN** plex to **STALE**. The next time the volume is started, the plex will be re-attached automatically.

cp The **fsgen** and **gen** usage types do not add any specific restrictions to the **cp** operation.

mv If the destination plex has unmapped regions (a range of blocks in the plex with no backing subdisk) that are not mapped in the source plex, or if the destination plex is shorter than the source plex, then the **–f** option is required. Even with **–f**, the operation will prevent the plex from being sparsed such that the volume would be left with two or more sparse, enabled, read-write plexes, but no complete plexes.

RAID5 USAGE-TYPE

The **raid5** usage type provides the following options as arguments to **–o** in addition to the required options:

force Force an operation that the Volume Manager considers potentially dangerous or of questionable use. This applies to attempts to dissociate the RAID-5 plex of a non-**EMPTY** volume or to remove the last RAID-5 logging plex of a non-**EMPTY** volume.

As with other usage types, if a **vxplex** operation is interrupted by a signal, then an attempt is made to restore the disk group configuration to a state that is roughly equivalent to its original state. If this attempt is interrupted (such as through another signal) then the user may need to perform some cleanup. The specific cleanup actions that are needed are written to the standard error before **volplex** exits.

The **raid5** usage type supports only the following keywords:

att Attaches the named plexes to the named volume. If a plex has a layout of **RAID**, the plex will be attached as the RAID-5 plex of the RAID-5 volume. To attach a RAID-5 plex to the volume, the volume must be disabled and be in the **EMPTY** state, and the RAID-5 plex will be given a state of **EMPTY**.

If a plex has a layout other than **RAID**, the plex will be attached as a RAID-5 logging plex for the RAID-5 volume. If the volume has no RAID-5 logging plexes, the log length for the volume will be set to the length of the smallest logging plex being attached. If the volume already has at least one logging plex, a plex can only be attached as a logging plex if it's contiguous length is at minimum the volume's logging length. RAID-5 logging plexes cannot be sparse in respect to the volume's logging length; attempts to attach a sparse logging plex will fail.

If the RAID-5 volume is not enabled, logging plexes are attached and marked as STALE. If the RAID-5 volume is enabled and has no logging plexes, attaching a logging plex will cause plexes being attached as logging plexes to be zeroed before they are enabled; otherwise, the new logging plexes are attached write-only and the contents of the existing logging plexes are copied to the new logging plexes using **ATOMIC_COPY** ioctls, after which the logs are enabled.

dis Dissociates the named plex from the RAID-5 volume to which it is attached. If the plex is the RAID-5 plex of the volume and the volume is not **EMPTY**, this requires the **–o force** option, as any data on the volume would be lost. If the plex is a logging plex for the volume and would leave the RAID-5 volume with no usable logging plexes, the **–o force** option is required.

Note that the RAID-5 usage type does not support the **det**, **cp**, **snapstart**, **snapdone** or **copy** keywords; these operations are either inappropriate or impossible to perform within the operational conecepts of RAID-5.

FILES

/etc/vx/type/*usetype***/vxplex**
The utility that performs **vxplex** operations for a particular volume usage type.

/etc/vx/type/fsgen/fs/*fstype***/vxsync**
Path to a program used with the **fsgen** usage type for synchronizing in-memory file system data with a volume, for the file system type *fstype*. The program is given arguments of a volume name and one or more plex names. For the **ufs** and **s5** file system types, this is a link to **sync**. For the **vxfs** usage type, this program uses the VxFS file system freeze feature to ensure a perfect synchronized detach.

EXIT CODES

The **vxplex** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered but rather denotes the first condition that prevented further execution of the utility. See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO **sync**(1M), **vxintro**(1M), **vxassist**(1M), **vxedit**(1M), **vxmend**(1M), **vxvol**(1M).

NAME

vxprint – display records from the Volume Manager configuration

SYNOPSIS

vxprint [**–AvpsdGhnlafmtqQ**] [**–g** *diskgroup*] [**–e** *pattern*] [**–D** *database*] [**–F** [*type:*]*format-spec*] [*name ...*]
vxprint [**–SAq**] [**–g** *diskgroup*]

DESCRIPTION

The **vxprint** utility displays complete or partial information from records in Volume Manager disk group configurations. Records can be selected by name or with special search expressions. Additionally, record association hierarchies can be displayed in an orderly fashion so that the structure of records is more apparent.

Dashes (–) are displayed in the output wherever there is no applicable record value.

The default output format is single-line records containing a record type, name, usage-type (or – for plexes and subdisks), enabled state (or – for subdisks), length, and a comment. A one-line header is written before any record information.

The **vxprint** utility can display disk group, disk media, volume, plex, and subdisk records. It cannot display disk access records. Use the **vxdisk list** operation to display disk access records, or physical disk information.

By default, all records in the **rootdg** disk group are displayed. Subdisks are sorted primarily by the subdisk device, secondarily by the device offset. Plex and volume records are sorted by name.

OPTIONS

The following options are recognized:

–S Display configuration summary information. The output consists of a header line followed by a line containing the total number of subdisks, plexes and volumes; the number of unassociated subdisks; and the number of unassociated plexes.

–A Print records from all active (imported) disk groups. Each disk group represented in the output is separated from other disk groups by blank lines. Also, a short header line introduces each disk group.

–g *diskgroup*
Select records from the specified disk group. The *diskgroup* option argument can be either a disk group name or disk group ID.

–v Select only volumes for display. This restricts the records matched with search patterns. Also, the default selection of all records in the database is then restricted to all volumes. If a *name* operand names a plex or subdisk, then a diagnostic is written to the standard error.

–p Select only plexes.

–s Select only subdisks.

–d Select only disk media records.

–G Select only disk group records.

Note: the **–v**, **–p**, **–s**, **–d**, and **–G** options may be combined to specify that more than

one record type is allowed. Specifying all options restores the default behavior. Also, note that without the addition of the **–A** option, the **–G** option will print at most one disk group record.

–e *pattern*
Use a volume configuration search expression to select records to be displayed. See **vol_pattern**(3X) for a description of search patterns.

–h List complete hierarchies below selected records. For volumes, this list includes all associated plexes and subdisks. For plexes, this list includes all associated subdisks. Hierarchies are separated in the output by a blank line. The order of output is a volume name, followed by one associated plex, followed by all of the subdisks for that plex, followed by another associated plex, followed by all of the subdisks for the second plex, and so on.

The **–v**, **–p**, and **–s** options limit the selection only of the head of a hierarchy. They do not prevent the display of associated records through the **–h** option.

Unless objects are named explicitly with *name* operands, a record is never displayed in two separate hierarchies. Thus, a selected plex is not displayed as a separate hierarchy if the volume that is associated with the plex is also selected.

–n Display only the names of selected records.

–l Display all information from each selected record. This information is in a free format that is not intended for use by scripts. The format is more convenient than the **–m** format for looking at records directly, because the density of information is more appropriate to human viewing.

–a Display all information about each selected record, one record per line. The format is the same as for the **–m** option, except that the **–a** option format appears on a single line with one space character between each field, and the list of associated records is not displayed. This format is useful for processing output through filters such as **sed** and **grep** that operate exclusively on one-line records, although the fields are not readily distinguishable. It isn't a practical format from the viewpoint of human readability.

–m Display all information about each selected record in a format that is useful as input to both the **vxmake** utility and to **awk**(1) scripts. The format used is the **vxmake** description format (see **vxmake**(4)). In addition to record information, the list of plex or subdisk records associated with selected volume or plex records is displayed. Each field is output on a separate line, indented by a single tab. Values for fields that contain comment-style strings are always preceded by one double-quote character.

–f Display information about each record as one-line output records containing the following fields:

1. Record type
2. Record name
3. Usage-type, volume association, or plex association (or – for unassociated plexes and subdisks)

4. Enabled state (or – for subdisks, disks, or disk groups)
5. Length, in units of system sectors
6. Plex association offset (or – for volumes, plexes, disks, or disk groups). This field will appear as **LOG** for log subdisks.
7. Usage-dependent state (or – for subdisks). If an exception condition is recognized (a plex I/O failure, removed or inaccessible disk, or an unrecovered stale data condition), then that condition is listed instead of any usage-type-dependent state.
8. The **tutil[0]** field. This field is set by usage-types as a lockout mechanism.
9. The **putil[0]** field. This field can be set to prevent associations of plex or subdisk records.

A one-line header is written before any record information.

–t Print single-line output records that depend upon the configuration record type. For disk groups, the output consists of the record type, **dg**, the disk group name, and the disk group ID.

For disk media records, the output consists of the following fields, in order from left to right:

1. Record type, **dm**
2. Record name
3. The underlying disk access record
4. The disk access record type (**sliced**, **simple**, or **nopriv**)
5. The length of the disk's private region
6. The length of the disk's public region
7. The path to use for accessing the underlying raw disk device for the disk's public region.

For subdisks, the output consists of the following fields, from left to right:

1. Record type, *sd*
2. Record name
3. Associated plex, or – if the subdisk is dissociated
4. The name of the disk media record used by the subdisk
5. Device offset in sectors
6. Subdisk length in sectors
7. Plex association offset optionally preceded by subdisk column number for subdisks associated to striped plexes, *LOG* for log subdisks, or the *putil[0]* field if the subdisk is dissociated. The *putil[0]* field can be non-empty to reserve the subdisk's space for non-volume uses. If the *putil[0]* field is empty, then print – for dissociated subdisks.
8. A string representing the state of the subdisk, one of **ENA** if the subdisk is usable; **DIS** if the subdisk is disabled; **RCOV** if the subdisk is part of a RAID-5 plex and has stale content; **DET** if the subdisk has been detached; **KDET** if the subdisk has been detached in the kernel due to an error; **RMOV** if the media record on which the subdisk is defined has been removed from

it's disk access record by a utility; or **NDEV** if the media record on which the subdisk is defined has no access record associated.

For plexes, the output consists of the following fields, from left to right:

1. Record type, **pl**
2. Record name
3. Associated volume, or – if the plex is dissociated
4. Plex kernel state
5. Plex utility state. If an exception condition is recognized on the plex (an I/O failure, removed or inaccessible disk, or an unrecovered stale data condition), then that condition is listed instead of the value of the plex record's **state** field.
6. Plex length in sectors
7. Plex layout type
8. Number of columns and plex stripe width, or – if the plex is not striped.
9. The plex I/O mode, either **RW** (read-write), **WO** (write-only), or **RO** (read-only).

For volumes, the output consists of the following fields, from left to right:

1. Record type, **v**
2. Record name
3. Associated usage type
4. Volume kernel state
5. Volume utility state
6. Volume length in sectors
7. The volume read policy.
8. The preferred plex, if the read-policy uses a preferred plex.

A header line is printed before any record information, for each type of record that could be selected based on the **–v**, **–p**, **–s**, and **–h** options. These header lines are followed by a single blank line.

–q Suppress headers that would otherwise be printed for the default and the **–t** and **–f** output formats.

–Q Suppress the disk group header that separates each disk group. A single blank line still separates each disk group.

–F [**type**:]*format_spec*

Set a literal format string to use for displaying record information. If the option argument begins with a comma-separated list of zero or more record types (**sd**, **plex**, or **vol**), followed by a colon, then the *format_spec* after the colon is used when printing the indicated record types. If no record types are specified, then all record types are assumed.

The order of **–F** options is significant, with specifications later in the option list overriding earlier specifications. Any use of **–F** overrides any other option letter specifying a type of format for the indicated record types. Thus, **–F**

vol:*format_spec* can be used with the **–t** option to change the format used for volumes, while still using the **–t** format for plex and subdisk records.

The **format-spec** string consists of literal text with embedded configuration record variables. Configuration record variables are introduced with a percent sign (**%**). The percent sign is followed by a variable name or by a variable name and optional field width in braces. The following formats are allowed for a variable specification:

%*field_name*
%{*field_name***}**
%{*field_name***:**[[–]*width*][*****]**}**
%{*field_spec***|***field_spec*[**|** ...]**}**

The first format specifies the exact field name.

The second format allows a field to be specified with immediately surrounding text that would otherwise be taken as part of the field name. The third format allows the specification of a justification and a field width. The fourth format allows alternate specifications to be used, either with or without justification and width specifications. For the fourth, the first specification is used if the specified field name is applicable to the record and is non-empty; otherwise, the next available specification is used. Any number of alternate specifications can be used.

If no field width is specified, then the number of output column positions used for the field is the smallest possible to contain the value; otherwise spaces are added in the output to make it *width* columns in length. A field is not truncated if the minimum number of column positions necessary for a value is greater than *width*.

If a field width is specified with a leading – character, then an output field is lengthened by adding spaces after the field value, yielding a left-justified field. Otherwise, spaces are added before the value, yielding a right-justified field.

If a field width is followed or replaced by an asterisk character, then an unrecognized or inappropriate field yields either no output for the field or a field containing all blanks. Without the asterisk, the printed field contains the character –.

One % can be displayed by including two % characters in *format_spec*.

See the ''Record Fields'' section for a description of the field names that can be specified. An invalid format string may yield unexpected output but does not generate an error.

–D *database*

Get a configuration from the specified location. The *database* option argument can be one of:

vxconfigd
(Default). Get a configuration from the volume configuration daemon.

–
Read a configuration from the standard input. The standard input is expected to be in standard **vxmake** input format.

RECORD FIELDS

The field names that can be used with the *format_spec* string of the **–F** option and that are produced for the **–m** or **–a** options are the same as those that can be provided as input to the **vxmake** utility. For a list of these field names, see **vxmake**(4). Some additional pseudo fields are also supported. These are:

name The name of the record being displayed. Because the record name is specified positionally within **vxmake** description formats, the **vxmake** utility and the **–m** and **–a** options to **vxprint** do not explicitly provide this field name.

rec_type or **rtype**
This is either **dg**, **dm**, **vol**, **plex**, or **sd**, depending upon whether the record being displayed is a disk group, disk, volume, plex, or subdisk, respectively.

short_type or **type**
This is either **dg**, **dm**, **v**, **pl**, or **sd**, depending upon the record type. This pseudo variable can be used in a 2-character field, if a full 4 character field (required by **rec_type**) is too large.

dgname or **dg_name**
This is the name of the disk group containing the record.

assoc The name of the volume or plex to which a plex or subdisk record is associated. If the record is not associated, this field is empty.

use_assoc
The usage type for volume records and the association name for associated plexes and subdisks. For dissociated plexes and subdisks, this is an empty string.

aslist A comma-separated list of subdisks or plexes that are associated with a plex or volume record.

sdaslist
A comma-separated list of subdisks associated with a plex. Each subdisk name is followed by a colon and the subdisk's plex association offset, in sectors. For volume records, this field is equivalent to **aslist**.

admin_state
The persistent state for a plex or volume record, accounting for any exceptional conditions. For volume records, this displays the **state** field. For plex records, this displays one of the following in the given precedence order: **NODEVICE** if an expected underlying disk could not be found; **REMOVED** if an underlying disk is in the **removed** state; **IOFAIL** if an unrecovered I/O failure caused the plex to be detached; **RECOVER** if a disk replacement left the plex in need of recovery, either from another plex or from a backup.

vname or **v_name**
The name of an associated volume record. For a volume record, this is the volume's name; for a plex record, this is the associated volume's name (if any); for a subdisk record, this is the associated volume of the associated plex (if any).

plname or **pl_name**
The name of an associated plex record. For a plex record, this is the plex's name; for a subdisk record, this is the associated plex's name (if any).

ncolumn_st_width
: For a striped plex, the number of columns and stripe width for a plex separated by a */*, or – if the plex is not striped.

column_pl_offset
: For a subdisk associated with a striped plex, the column number and column offset of the subdisk separated by a */*, or the plex offset if the subdisk is associated in a non-striped plex, or – if the plex is not associated.

sd_Pflag
: For a subdisk associated with a Raid-5 plex, this will display flags relating to the status of the subdisk. A **S** indicates that the subdisk is considered to contain stale data. A **d** indicates that the subdisk has been detached from the Raid-5 plex.

Displaying a boolean value always yields **on** or **off**. If a field containing a length or offset is specified in a *format_spec* string, then the result is the length or offset in sectors. When the field is displayed with **–m** or **–a**, the length or offset is displayed in sectors with a suffix of **s**.

EXIT CODES

The **vxprint** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered but rather denotes the first condition that prevented further execution of the utility.

See **vxintro**(1M) for a list of standard exit codes.

EXAMPLES

To display all records in all disk groups, with clearly displayed associations and with output lines tailored to each record type, use the command:

vxprint -Ath

If you tire of looking at the 5-line header and the extra disk group headers generated by this command, then you can remove all the headers by adding a **–q**.

To display all subdisks and all disk groups, in sorted order by disk, use:

vxprint -AGts

If all plexes are named based on volumes, this can be a convenient means of viewing large configurations. The association field for each of the subdisks names the plex, and the plex name will normally imply a volume association by the form of the plex name.

To display the names of all unassociated plexes, use the command

vxprint -n -p -e !assoc

To print all subdisks, including the subdisk name and either the subdisk plex association offset or the **putil0** field for dissociated subdisks, use:

vxprint -s -F "%{name:-14} %{pl_offset|putil0}"

SEE ALSO

awk(1), **grep**(1), **vxintro**(1M), **vxinfo**(1M), **vxmake**(1M), **vxmake**(4), **sed**(1).

NAME

vxreattach – reattach drives that have once again become accessible

SYNOPSIS

/etc/vx/bin/vxreattach [**–rb**] [*accessname* ...]
/etc/vx/bin/vxreattach **–c** *accessname*

DESCRIPTION

The **vxreattach** utility will, if possible, reattach disks to the disk group they were previously in and retains the same media name that they had.

This operation may be necessary if a disk has a transient failure, or if the Volume Manager is started with some disk drivers unloaded and unloadable. Disks will then enter the failed state. If the problem is fixed, then it may be possible to use the **vxreattach** command to reattach the disks without plexes being flagged as stale, as long as the reattach happens before any volumes on the disk are started.

The **vxreattach** command is called as part of disk recovery from the **vxdiskadm** menus.

vxreattach tries to find a disk in the same disk group with the same disk ID for the disk(s) to be reattached. Note that the reattach operation may fail even after finding the disk with the matching disk ID if the original cause (or some other cause) for the disk failure still exists.

The **vxreattach** utility has the following options:

-r This option causes the **vxreattach** utility to attempt to recover stale plexes of any volumes on the failed disk. It does this by calling **vxrecover**.

-b This option causes the reattach operation to be performed in the background.

-c This option checks whether a reattach is possible. No operation is performed, but the name of the disk group and disk media name at which the disk can be reattached will be output.

EXIT CODES

A zero exit status will be returned if it is possible to perform a reattach, non-zero otherwise.

SEE ALSO

vxintro(1M), **vxdiskadm**(1M), **vxrecover**(1M).

NAME

vxrecover – perform volume recovery operations

SYNOPSIS

vxrecover [**–g** *diskgroup*] [**–sbnpvV**] [**–o** *options*] [*volume* | *medianame* ...]

DESCRIPTION

The **vxrecover** program performs plex attach, RAID-5 subdisk recovery, and resynchronize operations for the named volumes, or for volumes residing on the named disks. If no *medianame* or *volume* operands are specified, then the operation applies to all volumes (or to all volumes in the specified disk group). If **–s** is specified, then disabled volumes will be started. With **–s** and **–n**, volumes are started, but no other recovery takes place.

Recovery operations will be started in an order that prevents two concurrent operations from involving the same disk. Operations that involve unrelated disks will run in parallel.

Options that can be specified to **vxrecover** are:

–s Start disabled volumes that are selected by the operation. Volumes will be started before any other recovery actions are taken. Volumes will be started with the **–o delayrecover** start option. This requests that any operations that can be delayed in starting a volume will be delayed. In other words, only those operations necessary to make a volume available for use will occur. Other operations, such as mirror resynchronization, attaching of stale plexes and subdisks, and recovery of stale RAID-5 parity will normally be delayed.

–b Perform recovery operations in background. With this option, **vxrecover** will put itself in background to attach stale plexes and subdisks, and to resynchronize mirrored volumes and RAID-5 parity. If this is used with **–s**, then volumes will be started before

–n Do not perform any recovery operations. If used with **–s**, then volumes will be started, but no other actions will be taken. If used with **–p**, then the only action of **vxrecover** will be to print a list of startable volumes.

–p Print the list of selected volumes that are startable. For each startable volume, a line is printed containing the following information: the volume name, the disk group ID of the volume, the volume's usage type, and a list of state flags pertaining to mirrors of the volume. State flags include the following: **kdetach** indicates that one of the mirrors was detached by an I/O failure; **stale** means that one of the mirrors needs recovery, but that the recovery is related to an administrative operation, not an I/O failure; **stopped** is printed if neither **kdetach** nor **stale** is appropriate for the volume.

–v Display information about each task started by **vxrecover**. For recovery operations (as opposed to start operations) print a completion status when each task completes.

–V Display each command executed by **vxrecover**. The ouput generated is more verbose than the job descriptions printed with the **–v** option.

–g *diskgroup*
Limit operation of the command to the given disk group, as specified by disk

group ID or disk group name. If no *volume* or *medianame* operands are given, then all disks in this disk group will be recovered; otherwise, the *volume* and *medianame* operands will be evaluated relative to the given disk group.

Without the **–g** option, if no operands are given, then all volumes in all imported disk groups will be recovered; otherwise, the disk group for each *medianame* operand will be determined based on name uniqueness within all disk groups.

–o *options*

Pass the given option arguments to the **–o** options for the **vxplex att** and **vxvol start** operations generated by **vxrecover**. An option argument of the form *prefix*:*options* can be specified to restrict the set of commands that the **–o** option should be applied to. Defined prefixes are: **vol** applies to all invocations of the **vxvol** utility (volume starts, mirror resynchronizations, RAID-5 partity rebuilds, and RAID-5 subdisk recoveries); **plex**, applies to all invocations of the **vxplex** utility (currently used only for attaching plexes); **attach** applies specifically to plex attach operations; **start** applies specifically to volume start operations; **recover** applies to subdisk recoveries; and **resync** applies to mirror resynchronization and RAID-5 parity recovery.

EXAMPLES

To recover, in background, any detached subdisks or plexes that resulted from replacement of a specified disk, use the command:

vxrecover -b *medianame*

If you want to see what's going on, for example, to know when the operations are complete, and to get an idea of the progress, use the command:

vxrecover -v *medianame*

SEE ALSO

vxintro(1M), **vxplex**(1M), **vxvol**(1M).

NAME

vxresize – change the length of a volume containing a file system

SYNOPSIS

/etc/vx/bin/vxresize [**–g** *diskgroup*] [**–F** *fstype*] [**–xsb**] *volume new_length* [*medianame* ...]

DESCRIPTION

The **vxresize** command will either grow or shrink both the file system and its underlying volume to match the specified new volume length. The ability to grow or shrink is file system dependent. Some file system types may require that the file system be unmounted for the operation to succeed.

In some situations, when resizing large volumes, this command may take some time to complete.

The *new_length* operand can begin with + or – to indicate that the new length should be computed by adding to or subtracting from the current volume length.

The *medianame* operands name disks to use for allocating new space for a volume. These arguments can be either a simple name for a disk media record, or they can be of the form *medianame,offset* to specify an offset within the named disk. If an offset is specified, then regions from the that offset to the end of the disk are considered candidates for allocation.

The **vxresize** command has the following options:

–g *diskgroup*
: Limit operation of the command to the given disk group, as specified by disk group ID or disk group name. The *volume_name* operand will be evaluated relative to the given disk group.

–x
: Require that the operation represent an increase in the volume length. Fail the operation otherwise.

–s
: Require that the operation represent a decrease in the volume length.

–b
: Cause the resize operation to be performed in the background. The command will return quickly but the resize will be in progress. Use the **vxprint** command to determine when the operation has completed.

–F *fstyp*
: This option supplies the type of the file system to be resized.

FILES

/ etc / vx / type / *usage_type* / vxresize
: The name of the command that will perform the resize depending on the usage type of the volume.

SEE ALSO

vxintro(1M), **vxprint**(1M).

NAME

vxrootmir – mirror areas necessary for booting to a new disk

SYNOPSIS

/etc/vx/bin/vxrootmir *medianame*

DESCRIPTION

The **vxrootmir** script creates mirrors of volumes required in booting. It creates mirrors for **rootvol**, **swapvol**. It will also create mirrors for **usr** and **var** volumes if they exist. The mirrors will be created on the specified disk media device. The subdisks for these volumes will be on cylinder boundary and partitions will be created for them.

The specified disk media device should have enough space to contain the mirrors for **rootvol**, **swapvol**, **usr** and **var** else it will fail. Also the slices 0 and 1 and 2 other slices must be free. 0 and 1 slices are used to create partitions for **root** and **swap** respectively. The other 2 slices are required for **usr** and **var**.

All disk regions required for booting will be set up by invoking the **installboot** command and partitions for the new volume mirrors will be created.

This script is called by the **vxmirror** command if the root disk is required to be mirrored. It is also called from the **vxdiskadm** menus through the choice of the **mirror volumes on a disk** operation.

SEE ALSO

installboot(1M), **vxintro**(1M), **vxdiskadm**(1M), **vxmirror**(1M).

NAME

vxsd – perform Volume Manager operations on subdisks

SYNOPSIS

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–l** *offset*] **assoc** *plex subdisk ...*

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–l** *column[/offset]*] **assoc** *plex subdisk*[*:column*[*/offset*]] ...

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] **aslog** *plex subdisk*

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–p** *plex*] **dis** *subdisk ...*

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–p** *plex*] **–s** *size* **split** *subdisk newsd* [*newsd2*]

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–p** *plex*] **join** *sd1 sd2 ... newsd*

vxsd [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] [**–v** *volume*] [**–p** *plex*] **mv** *oldsd newsd* [*newsd ...*]

DESCRIPTION

The **vxsd** utility performs Volume Manager operations on subdisks and on plex-and-subdisk combinations. The first operand is a keyword that determines the specific operation to perform. The remaining operands specify the configuration objects to which the operation is to be applied.

Each operation can be applied to only one disk group at a time, due to internal implementation constraints. Any plex or subdisk name operands will be used to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). A specific disk group can be selected with **–g** *diskgroup*.

If a **vxsd** operation is interrupted by a signal, then an attempt is made to restore the disk group configuration to a state that is roughly equivalent to its original state. If this attempt is interrupted, such as through another signal, then the user may need to perform some cleanup. The specific cleanup actions that are needed are written to the standard error before **vxsd** exits.

These are the recognized operation keywords:

assoc Associate each named subdisk operand with the specified plex. The first form applies to concatenated plexes (i.e., with a layout of **concat**). The offset within the plex for the association can be specified with *–l*, which takes a standard volume manager length number (see *vxintro(8)*). If no offset is specified, then the default is to associate the subdisk at the end of the plex, thus extending the length of the plex by the length of the new subdisk.

For striped plexes a column number for the subdisk association may be specified. The offset is interpreted as the column offset for the subdisk. If only one number is specified with *–l* for striped plexes, the number is interpreted as a column number and the subdisk is associated at the end of the column.

The column or column and offset at which a subdisk is to be associated can also be specified as part of the subdisk name in the same manner as subdisks associations are specified for plex creations in **volmake** (see **volmake**(4)). When

specifying multiple subdisks, if no column or column / offset is specified for a subdisk, it is associated after the previous subdisk.

A subdisk cannot be associated to overlap with an another associated subdisk in the same plex.

If the named plex is associated with a volume, then the rules for performing the operation depend upon the usage type of the volume. A subdisk cannot be associated to a plex if the **putil0** field for the subdisk is not empty. Creating a subdisk with the **putil0** field set to a non-empty value is a sufficient means of ensuring that no Volume Manager operation will write to the region of disk blocks allocated to the subdisk because the subdisk cannot be associated through any means to a plex, and because subdisks cannot be used directly to read from or write to a disk.

aslog Associate the named subdisk with the named plex as a log area for the plex. At most one log subdisks can be associated with a plex at any one time. At the present time, log subdisks can be used only with the dirty-region-logging feature, as defined by the *DRL* volume logging type. If the named plex is associated with a volume, then the rules for performing the operation depend upon the usage type of the volume. A subdisk cannot be associated if the *putil0* field is set on the subdisk, just as with *vxsd assoc*.

dis Dissociate each specified subdisk from the plex that it is associated with. If a subdisk is associated (through its plex) with a volume, then the rules for performing the operation depend upon the usage type of the volume.

Subdisk dissociation can be used as part of tearing down a plex, or as part of reorganization of disk space usage. Typically, the subdisk is no longer needed after dissociation. To support this type of use, **–o rm** can be specified to remove the named subdisks after successful dissociation.

split Split the subdisk *subdisk* into two subdisks that reside on the same section of the same device, and that have contiguous plex associations (if the named subdisk is associated). The first of the two resultant subdisks will have a length of *size*, and the second will take up the remainder of the space used by the original subdisk. If both *newsd* and *newsd2* are specified, then the resultant subdisks are *newsd* and *newsd2*. If no *newsd2* operand was specified, then the resultant subdisks are named *subdisk* and *newsd*.

If the named subdisk is associated with an associated plex, then the rules for performing the operation depend upon the usage type of the volume. Log subdisks cannot be split.

join Join the subdisks named by the *sd* operands to form a new subdisk named *newsd*. The *sd* operands must specify subdisks that represent contiguous sections of the same device, and of the same plex (if they are associated). For a striped plex the *sd* operands must be in the same column. At least two *sd* operands are required. At the end of the operation, the *sd* configuration objects are removed. The *newsd* operand can have the same name as one of the *sd* operands, or it can have a different name.

If the *sd* operands are associated with an associated plex, then the rules for performing the operation depend upon the usage type of the volume.

mv Move the contents of *oldsd* onto the new subdisks and replace *oldsd* with the new subdisks for any associations. If multiple new subdisks are specified, they are associated starting where the old subdisk began and placed consecutively with no space between them. The operation requires that *oldsd* be associated with an associated plex and that all new subdisks be dissociated. The operation can be used on a subdisk that is used by an active volume, and will ensure that data is copied and associations are changed without loss or corruption of data. The rules for performing the operation depend upon the usage type of the volume.

Moving a subdisk is the normal means of reorganizing disk space. For example, move regions of disk used by one volume to another disk to reduce contention on the original disk. Typically, once the operation completes, the original subdisk is no longer needed and can be removed. To support this use of the operation, **−o rm** can be specified to remove *oldsd* after successful completion of the operation.

OPTIONS

The following options are recognized:

−g *diskgroup* Specify the disk group for the operation, either by disk group ID or by disk group name. By default, the disk group is chosen based on the *name* operands.

−U *usetype* Limit the operation to apply to this usage type. Attempts to affect volumes with a different usage type will fail.

−o *useopt* Pass in usage-type-specific options to the operation. A certain set of operations are expected to be implemented by all usage types:

slow[**=***iodelay*]
Reduce the system performance impact of copy operations. Copy operations are usually a set of short copy operations on small regions of the volume (normally from 16 kilobytes to 128 kilobytes). This option inserts a delay between the recovery of each such region. A specific delay can be specified with *iodelay* as a number of milliseconds, or a default is chosen (normally 250 milliseconds).

iosize=*size*
Perform copy operations in regions with the length specified by *size*, which is a standard Volume Manager length number (see **vxintro**(1M)). Specifying a larger number typically causes the operation to complete sooner, but with greater impact on other processes using the volume. The default I/O size is typically 32 kilobytes.

rm Remove the subdisks after successful completion of a **vxsd dis** operation. Remove the source subdisk after successful completion of **vxsd mv**.

–V Write a list of utilities that would be called from **vxsd**, along with the arguments that would be passed. The **–V** performs a ''mock run'' so the utilities are not actually called.

–s *size* Specify the size for the subdisk **split** operation. This option takes a standard Volume Manager length number (see **vxintro**(1M)).

–l [column/]offset Specify the offset of a subdisk within a plex address space for the *vxsd assoc* operation. For striped plexes a column number may be optionally specified. If one number is given for striped plexes the number is interpreted as a column number and the subdisk is associated at the end of the column. The offset is a standard volume manager length number (see *volintro(8)*).

–v *volume* Require that a named plex be associated with this volume, or that a named subdisk (source subdisk for **vxsd split**, **join**, and **mv**) be associated with a plex that is associated with this volume.

–p *plex* Require that a named subdisk (source subdisk for **vxsd split**, **join**, and **mv**) be associated with this plex.

–f Force an operation that the Volume Manager considers potentially dangerous or of questionable use. This permits a limited set of operations that would otherwise be disallowed. Some operations may be disallowed even with this flag.

FSGEN AND GEN USAGE-TYPES

The **fsgen** and **gen** usage types provide identical semantics for all operations of the **vxsd** utility.

In addition to the standard **–o** options required for all usage types, the **fsgen** and **gen** usage types provide the following additional option:

force Force an operation that the Volume Manager considers potentially dangerous or of questionable use. This applies to attempts to dissociate subdisks (making a plex sparse) and to attempts to move subdisks onto subdisks that have a different size. This flag is the same as **–f**.

Limitations and extensions for the **fsgen** and **gen** usage types consist of the following:

assoc If the named plex is enabled, and is associated with an enabled plex, then the named plex must **ACTIVE** or **EMPTY**. Subdisks can be associated with a non-enabled plex only if the utility state of the plex is **EMPTY**, **STALE**, or **OFFLINE**, or if the plex is **CLEAN** and no other plexes associated with the volume are **CLEAN** or **ACTIVE**.

If the subdisk is associated with a non-enabled plex, or if it is associated with the only enabled, read-write plex in a volume, then the operation completes without copying any data onto the subdisk. If the subdisk is associated with an enabled plex in a mirrored volume, then the operation may have to copy data from the volume onto the new subdisk before the operation can complete.

aslog If a log subdisk is associated with a plex that is associated with a volume that has a logging type of *UNDEF*, then the logging type of the volume is converted to

DRL. Logging of volume changes is not enabled until there are at least two read-write mode plexes attached to the volume.

dis Dissociating a subdisk requires use of **–f** if it would cause an enabled plex in an enabled volume to become sparse relative to the volume. Even with **–f**, it is not possible to make two plexes sparse if no complete, enabled, read-write plexes would remain associated. For disabled volumes, a similar check is made with respect to **ACTIVE** and **CLEAN** plexes.

split and **join**
The **fsgen** and **gen** usage types apply no additional restrictions and adds no extensions to the **split** and **join** operations.

mv If the total size of the destination subdisks differs from that of the source subdisk, then the **–f** option must be specified. The operation still fails if the total size of the destination subdisks is larger than the source subdisk and if the address range of any destination subdisk would conflict with another subdisk that is associated with the plex. The total size of the destination subdisks cannot be larger than the source subdisk if the kernel state of the volume or plex is **detached**.

The operation fails if the total size of the destination subdisks is smaller than the source subdisk and the operation would cause the total number of complete, enabled, read-mode plexes in the volume to drop to zero, while leaving more than one sparse, enabled, read-write plex.

RAID-5 USAGE TYPE

In addition to the standard **–o** options required for all usage types, the **raid5** usage type provides the following additional options:

force Force an operation that the Volume Manager considers potentially dangerous or of questionable use. This applies to attempts to move a subdisk in a RAID-5 plex if the volume the plex is associated with does not have a log plex. This flag is the same as **–f**.

The **raid-5** usage type supports the following keywords:

assoc Associate the named subdisks with the named RAID-5 plex. If plex is enabled and is associated with an enabled volume, then any data that maps onto the subdisk will be regenerated from the other columns of the RAID-5 plex. This is done by marking the subdisk as **stale** and **writeonly**, regenerating the data via **VOL_R5_RECOVER** ioctls, and then turning off the **stale** and **writeonly** flags.

If the RAID-5 plex is not associated or the RAID-5 volume is not **ENABLED**, the subdisk is associated and marked as **stale**. The subdisk's contents will be recovered when the volume is started.

The **assoc** operation may not be used on a log plex.

dis Dissociate the named subdisks from the named RAID-5 plex. If removing the subdisk would make the volume unusable (because other subdisks in other columns at the same altitude are unusable or missing) and the volume is not disabled and **EMPTY**, the operation is not allowed. If the volume is disabled and

non-**EMPTY** the operation requires use of **–f**.

The **dis** operation may not be used on a log plex.

split and **join**

The **raid-5** usage type applies no additional restrictions and adds no extensions to the **split** and **join** operations. These operations may not be used on a log plex.

mv If the old subdisk is associated with RAID-5 plex that is associated to a RAID-5 volume, the volume must be enabled for the move operation to complete. The **mv** operation is not allowed if the volume has stale parity or has missing or stale subdisks at the same altitude as the subdisk being replaced.

The **mv** operation first dissociates the old subdisk and then associates the new subdisk in its place and then recovers the data using **VOL_R5_RECOVER** ioctls, as is done for the **assoc** operation. If the RAID-5 volume has no valid logs, the operation requires use of **–f**. This is necessary because if a crash were to occur while the data on the new subdisks was being recovered, the parity could become stale while some of the new subdisks are marked **stale** thus rendering the volume unusable.

The **mv** operation may be used on a log plex. Similar rules as those defined for the **fsgen** and **gen** usage type plexes apply to log plexes. The **force** or **-f** must be used to make a log plex sparse.

Note that there is no **aslog** operation for the **raid5** usage type. Logging is done on a plex level and therefore **vxsd aslog** is unneeded. Log plexes can be associated with RAID-5 volumes using the **vxplex att** command.

FILES

/etc/vx/type/*usetype***/vxsd**

The utility that performs **vxsd** operations for a particular volume usage type.

EXIT CODES

The **vxsd** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility. See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO

vxintro(1M), **vxplex**(1M), **vxsd**(1M), **vxvol**(1M).

NAME

vxserial – Volume Manager licensing key utility

SYNOPSIS

vxserial [**–cpt**]

DESCRIPTION

The Volume Manager licensing key utility, **vxserial**, is responsible for maintaining the Volume Manager license key file.

The **vxserial** utility has the following options:

–c Creates a license key file. **Vxserial** will prompt for a license key. This option should be used in most cases.

–p Print available Volume Manager licenses in a system.

–t *feature* Test a Volume Manager license in a system.

EXAMPLES

The following is an example of how to use **vxserial** to initialize the Volume Manager licensing key file:

```
vxserial -c
```

Here the **vxserial** command will prompt for a license key and create the license key file once the key is entered.

NAME

vxsparecheck – monitor the Volume Manager for failure events and replace failed disks

SYNOPSIS

/etc/vx/bin/vxsparecheck [*mail-address* ...]

DESCRIPTION

The **vxsparecheck** command monitors the Volume Manager by analyzing the output of the **volnotify**(1) command, waiting for failures to occur. It then sends mail via **mailx**(1) to the login(s) specified on the command line, or (by default) to *root*. It then proceeds to attempt the replacement of any failed disks. Once an attempt at replacement is complete, mail will be sent indicating the status of each disk replacement.

The mail notification that is sent when a failure is detected follows this format:

Failures have been detected by the VERITAS Volume Manager:

failed disks:
medianame
...
failed plexes:
plexname
...
failed subdisks:
subdiskname
...
failed volumes:
volumename
...

The Volume Manager will attempt to find hot-spare disks to replace any failed disks and attempt to reconstruct any data in volumes that have storage on the failed disk.

The *medianame* list specifies disks that appear to have completely failed. The *plexname* list show plexes of mirrored volumes that have been detached due to I/O failures experienced attempting to do I/O to subdisks they contain. The *subdiskname* list specifies subdisks in RAID-5 volumes that have been detached due to I/O errors. The *volumename* list shows non-RAID-5 volumes that have become unusable because disks in all of their plexes have failed (and are listed in the "failed disks" list) and shows those RAID-5 volumes that become unusable because of multiple failures.

If any volumes appear to have failed, the following paragraph will be included in the mail:

The data in the failed volumes listed above is no longer available. It will need to be restored from backup.

REPLACEMENT PROCEDURE

Once mail has been sent, **vxsparecheck** attempts to find a hot spare replacement for any disks that appear to have failed (i.e. those listed in the *medianame* list). This consists of finding an appropriate replacement the those eligible hot-spares in the same disk group

as the failed disk. A disk is eligible as a replacement if it it a valid Volume Manager disk, has been marked as a hot-spare disk and contains enough space to hold the data contained in all the subdisks on the failed disk.

To determine which disk from among the eligible hot spares should be used, **vxsparecheck** first consults the file */etc/vx/sparelist* (see below). If the file does not exist or lists no eligible hot spares for the failed disk, the disk that is "closest" to the failed disk is chosen. The value of "closeness" depends on the controller, target and disk number of the failed disk. A disk on the same controller as the failed disk is closer than a disk on a different controller; and a disk under the same target as the failed disk is closer than one under a different target.

If no hot spare disk can be found, the following mail is sent:

> **No hot spare could be found for disk** *medianame* **in** *diskgroup***. No replacement has been made and the disk is still unusable.**

The mail will then explain the disposition of volumes that had storage on the failed disk. The following message lists disks that had storage on the failed disk, but are still usable:

> **The following volumes have storage on** *medianame***:**
>
> *volumename*
>
> **These volumes are still usable, but the redundancy of those volumes is reduced. Any RAID-5 volumes with storage on the failed disk may become unusable in the face of further failures.**

If any non-RAID-5 volumes were made unusable due to the failure of the disk, the following message is included:

> **The following volumes:**
>
> *volumename*
>
> **have data on** *medianame* **but have no other usable mirrors on other disks. These volumes are now unusable and the data on them is unavailable.**

If any RAID-5 volumes were made unavailable due to the disk failure, the following message is included

> **The following RAID-5 volumes:**
>
> *volumename*
>
> **had storage on** *medianame* **and have experienced other failures. These RAID-5 volumes are now unusable and data on them is unavailable.**

If a hot-spare disk was found, a hot-spare replacement is attempted. This involves associating the device marked as a hot spare with the media record that was associated with the failed disk. If this is successful, the **vxrecover**(1M) command is used in the background to recover the contents of any data in volumes that had storage on the disk.

If the hot-spare replacement fails, the following message is sent:

> **Replacement of disk** *medianame* **in group** *diskgroup*
> *failed. The error is:*
>
> *error message*

If any volumes (RAID-5 or otherwise) are rendered unusable due to the failure, the following message is included:

> **The following volumes:**
>
> *volumename*
>
> **occupy space on the failed disk and have no other available mirrors or have experienced other failures. These volumes are unusable, and the data they contain is unavailable.**

If the hot-spare replacement procedure completed successfully and recovery is under way, a final mail message is sent:

> **Replacement of disk** *medianame* **in group** *diskgroup*
> **with disk device** *sparedevice* **has successfully completed and recovery is under way.**

If any non-RAID-5 volumes were rendered unusable by the failure despite the successful hot-spare procedure, the following message is included in the mail:

> **The following volumes:**
>
> *volumename*
>
> **occupy spare on the replaced disk, but have no other enabled mirrors on other disks from which to perform recovery. These volumes must have their data restored.**

If any RAID-5 volumes were rendered unusable by the failure despite the successful hot-spare procedure, the following message is included in the mail:

> **The following RAID-5 volumes:**
>
> *volumename*
>
> **have subdisks on the replaced disk and have experienced other failures that prevent recovery. These RAID-5 volumes must have their data restored.**

If any volumes (RAID-5 or otherwise) were rendered unusable, the following message is also included:

To restore the contents of any volumes listed above, the volume should be started with the command:

vxvol -f start <volume-name>

and the data restored from backup.

SPARELIST FILE

The sparelist file is a text file that specifies an ordered list of disks to be used as hot-spares when a specific disk fails. The system-wide sparelist file is located in **/etc/vx/sparelist**. Each line in the sparelist file specifies a list of spares for one disk. Lines beginning with the pound ('**#**') character and empty lines are ignored. The format for a line in the sparelist file is:

[*diskgroup***:]** *diskname* **:** *spare1* **[** *spare2* **...]**

The *diskgroup* field, if present, specifies the diskgroup within which the disk and designated spares reside. If not present, **rootdg** is presumed. *diskname* specifies the disk for which spares are being designated. The *spare* list after the colon lists the disks to be used as hot-spares. The list is order dependent; in case of failure of *diskname*, the spares are tried in order. A spare will be used only if it is a valid hot spare (see above). If the list is exhausted without finding any spares, the default policy of using the closest disk is used.

FILES

/etc/vx/sparelist

The file through which a list of disks can be explicitly specified to serve as hot spares for a certain disk.

NOTES

The sparelist file is not checked in any way for correctness until a disk failure occurs. It is possible to inadvertantly specify a non-existent disk or inappropriate disk or diskgroup. Malformed lines are also ignored.

SEE ALSO

mailx(1), **vxintro**(1M), **vxrecover**(1M).

NAME

vxstat – Volume Manager statistics management utility

SYNOPSIS

vxstat [**–g** *diskgroup*] [**-i** *interval* [**–c** *count*]] [**–f** *fields*] [**–r**] [**–psvd**] [*object ...*]

DESCRIPTION

The **vxstat** utility prints and resets statistics information on one or more volumes, plexes, subdisks, or disks.

The **vxstat** utility reads statistics from the volume device files in the directory **/dev/vx/rdsk/** and prints them to standard output. These statistics represent volume, plex, subdisk, and disk activity since boot time. If no *object* operands are given, then statistics from all volumes in the configuration database are reported. *object* can be the name of a volume, plex, subdisk, or disk.

OPTIONS

The following options are recognized:

–g *diskgroup*
Select records from the specified disk group. The *diskgroup* option argument can be either a disk group name or disk group ID.

-i *interval*
Print the change in volume statistics that occurs after every *interval* seconds. The first interval is assumed to encompass the entire previous history of *objects.* Subsequent displays will show statistics with a zero value if there has been no change since the previous interval.

-c *count*
Stop after printing interval statistics *count* times.

-f *fields* Select the display of statistics collected. The following options are available:

s Statistics on read and write operations. Displays six fields: the number of read operations, the number of write operations, the number of blocks read, the number of blocks written, the average time spent on read operations in the interval, and the average time spent on write operations in the interval. These statistics are displayed as the default output format.

a Statistics on atomic copies performed (has meaning only for mirrored volumes). Displays the number of operations, number of blocks, and the average time spent per operation.

v Statistics on verified reads and writes (has meaning only for mirrored volumes). Displays six fields: the number of verified read operations, the number of verified write operations, the number of blocks read, the number of blocks written, the average time spent on verified read operations in the interval, and the average time spent on verified write operations in the interval.

c Statistics on corrected (fixed) read operations (has meaning only for mirrored or RAID-5 volumes). Displays the number of fixed read and write operations. NOTE: Currently, only read operations are ever corrected so

the number of fixed writes will always be zero.

f Displays the number of failed read and write operations.

b Displays the statistics on read-writeback mirror consistency recovery operations (has meaning only for mirrored volumes). Displays three fields: the number of read-writeback operations, the number of blocks involved in read-writeback operations, and the average time for completing a read-writeback operation. While in recovery mode, most read operations to a mirrored volume invoke read-writeback consistency recovery.

F Statistics for full-stripe writes on a RAID-5 volume. The number of operations represents the number of write operations within a stripe that were conducted as a full-stripe write optimization. Full-stripe writes represent considerably less overhead than read-modify-writes both in terms of overall I/O time, latency and CPU overhead. The total number of blocks represents the total size of the written data and the average time is the time taken for a full-stripe write operation. Since the I/O may be larger then a single stripe, more than one stripe operation may be seen for a single logical I/O request.

M Read-modify-write statistics. Each operation represents a read-modify write operation performed within a stripe. I/O crossing a stripe boundary will be represented by more than a single read-modify write operation. The number of blocks counted represents only the size of the requested write. The read portion of the I/O can be derived.

W Reconstruct write statistics. Each operation counted is for a reconstruct write operation performed as an optimization of a write operation within a stripe. The number of blocks counted represents the count of data blocks written not including parity or read operations.

R Reconstruct read operations. Each operation is a seperate reconstruct read operation. A single stripe read or write operation can lead to numerous reconstruct read operations since each reconstruction takes place at the subdisk level. A detached column can consist of several subdisks each of which will lead to a reconstruct read operation.

0 Statistics for the **VOL_R5_ZERO** operation. Each operation represents one call to the **VOL_R5_ZERO** ioctl. The number of blocks is based on the number of zero'd blocks written to the array. The average time is the time taken to complete the entire ioctl operation.

S Statistics for the **VOL_R5_RESYNC** operation. Each operation count represents one call to the **VOL_R5_RESYNC** ioctl. The number of blocks represents the resulting number of blocks that were written to the parity regions as part of the resynchronization of parity. The read operations are not counted towards the total.

C Statistics for the **VOL_R5_RECOVER** operation. Each operation count represents one call to the **VOL_R5_RECOVER** ioctl. The number of

blocks represents the resulting number of blocks that were written to the missing column region as part of the data recovery. The read operations are not counted towards the total.

-r Reset statistics instead of printing them. This option will follow the same selection rules as printing for any type selection arguments or for any named objects. If an interval was specified on the command line, then the first set of statistics will not be printed since they will have been reset to zero. Subsequent activity will cause printing of statistics as normal.

-v Display statistics for volumes on the *objects* specified on the command line. For an *object* that is a plex or a subdisk, displays information about the volume with which the *object* is associated. If an *object* supplied is a disk, then any volumes that occupy any part of the disk will be selected.

-p Display statistics for plexes on the *object* specified on the command line. For subdisk *objects*, displays information about a plex with which it is associated.

-s Display statistics for subdisks on the *objects* specified on the command line.

-d Display statistics for disks on which the *object* specified on the command line is fully or partially located.

OUTPUT FORMAT

Summary statistics for each object are printed in one-line output records, preceded by two header lines. The output line consists of blank-separated fields for the object type, object name (standard), and the fields requested by the **-f** switch in the order they are specified on the command line.

If the **-i** *interval* option was supplied, then statistics will be prefaced with a time-stamp showing the current local time on the system.

EXIT CODES

The **vxstat** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility. See **vxintro**(1M) for a list of standard exit codes.

EXAMPLES

To display statistics for all subdisks associated with all volumes, use the command:

vxstat -s

To display statistics for the plexes and subdisks of a volume named **blop**, use the following:

vxstat -ps blop

To reset all statistics for a disk group named **foodg**, type the following command:

vxstat -g foodg -r

To display 5 sets of disk statistics at 10 second intervals, use the following:

vxstat -i 10 -c 5 -d

SEE ALSO **vxintro**(1M), **vxtrace**(7), and **vxtrace**(1M).

NAME

vxtrace – trace operations on volumes

SYNOPSIS

vxtrace [**–eEla**] [**–d** *outputfile*] [**–f** *inputfile*] [**–t** *timeout*] [**–c** *eventcount*] [**–w** *waitinterval] [–***b** *buffersize] [–***g** *diskgroup] [–***o** *objtype [,objtype*]...] [*name* | *device*] ...

DESCRIPTION

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

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

OPTIONS

The following options are recognized by **vxtrace**:

–e Select new error trace data. The default is to select I/O trace data.

–E Select pre-existing error trace data. This can be combined with **–e** to get both preexisting trace data and new trace data.

–l Long format. Print all available fields for all tracing records, rather than a subset of the available fields. The default is using short format.

–d *outputfile*
Write (dump) binary trace data to the specified output file.

–a Append to the *outputfile* rather than truncating it. By default, the output file is truncated.

–f *inputfile*
Read binary trace data from the specified input file, rather than from the kernel.

–t *timeout*
Accumulate trace data for at most *timeout* seconds and then exit.

–c *eventcount*
Accumulate at most *eventcount* events and then exit. The *timeout* and *eventcount* options can be used together.

–w *waitinterval*
If **vxtrace** waits for *waitinterval* seconds without receiving any new events, print "*waiting ...*" to allow scripts to wake up and process previously accumulated events. This is mostly of use for processing errors. The "*waiting ...*" message does not count as an event for the purposes of the **–c** option.

–b *buffersize*
Set the kernel I/O trace buffer size or set the read buffer size when used with the **–f** option. The Volume Manager kernel allocates a private kernel space to buffer the I/O trace records for each **vxtrace** command. The default buffer size is 8K bytes. Some trace records may be discarded if the trace buffer is too small. This option can be used to set a larger or a smaller kernel trace buffer size. The buffer size is

specified as a standard Volume Manager length [see **vxintro**(1M)]. Depending on the Volume manager kernel configuration, usually only a maximum of 64K bytes buffer size will be granted.

−g *diskgroup*
Select objects from the specified disk group. The disk group can be specified either by disk group ID or by disk group name. With no *name* or *device* operands, all appropriate objects in the disk group are selected. With *name* operands, *diskgroup* names the disk group that is expected to contain the named configuration record.

−o *objtype*[,*objtype*]...
Select object based on the *objtype* option arguments. Multiple types of objects can be specified with one or several **−o** options. The possible object selection types are:

v | vol | volume
Select mirrored or RAID-5 volume kernel objects.

p | pl | plex
Select striped or concatenated plex kernel objects.

m | mv | mirror
Select mirrored volume kernel objects.

s | sd | subdisk
Select subdisk kernel objects.

disk | physical
Select VxVM physical disks.

dev | logical
Select virtual disk devices.

logvol Select DRL or RAID5 log volumes.

logplex
Select RAID5 log plexes.

logsd Select DRL or RAID5 log subdisks.

log Select all log objects.

all | ALL
Select all possible virtual disk devices, kernel objects and physical disks.

If *name* or *device* operands are provided, then kernel objects of the requested types are selected if they are associated with the configuration records or virtual disk devices indicated by those operands.

OPERANDS

Operands specify configuration record names, or physical or virtual disk device nodes (by device path). If no object types were selected with the **−o** option, then only trace records corresponding to the indicated configuration records or devices are selected; otherwise, objects of the requested types are selected if they are associated in any way with the named configuration record or device.

If a *name* argument dose not match a regular configuration record but does match a disk access record, then the indicated physical disk is selected. Physical disks can also be selected by the device path of the public region partition, or by the disk media record name.

By default *name* arguments are searched for in all disk groups or in the disk group specified using the **–g** option. Without the **–g** option, a record that is found in more than one disk group will cause an error unless the record is in the **rootdg** disk group (in which case, the record in the **rootdg** disk group is selected). The disk group for any individual *name* operand can be overridden using the form:

diskgroup/recordname

NOTE: When reading trace data from a file with **–f** option, association information is not available.

EXAMPLES:

To trace all physical disk I/Os, use the command:

vxtrace -o disk

To trace virtual disk device I/Os to the device associated with volume **testvol** use either of the commands:

vxtrace -o dev testvol

vxtrace /dev/vx/dsk/testvol

To trace all log subdisks associated with volume **testvol** use the commands:

vxtrace -o logsd testvol

To trace all log objects, use the command:

vxtrace -o log

To accumulate 10 seconds worth of trace data for disk04 and then format that data, use:

vxtrace -t 10 -d /tmp/tracedata disk04

vxtrace -l -f /tmp/tracedata

To read error trace data into a script for processing, using 10 second pauses to generate mail messages, use the command:

vxtrace -leE -w 10 | while read ...

FILES

/dev/vx/trace

SEE ALSO

vxintro(1M), **vxstat**(1M), **vxtrace**(7)

NAME

vxunroot – remove Volume Manager hooks for rootable volumes

SYNOPSIS

/etc/vx/bin/vxunroot

DESCRIPTION

The **vxunroot** script causes the *root, swap, usr and var file systems to be converted back into being accessible directly through disk partitions instead of through volume devices. Other changes made to ensure the booting of the system from the root volume are also removed such that the system will boot with no dependency on the Volume Manager.*

For **vxunroot** to work properly, all but one plexes of *rootvol*, *swapvol*, *usr* and *var* should be removed. The plexes left behind for the above volumes should be the ones created by **vxrootmir** or the original ones created when the rootdisk was encapsulated. This will ensure that the underlying subdisks will have equivalant partitions defined for them on the disk. If none of these condifitons are met, the **vxunroot** operation will fail and none of the volumes will be converted to disk partitons.

The unwanted plexes can be removed using either **vxedit** or **vxplex**.

SEE ALSO

vxintro(1M), **vxdiskadm**(1M), **vxedit**(1M), **vxplex**(1M).

NAME

vxvol – perform Volume Manager operations on volumes

SYNOPSIS

vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **init** *init_type vxvol* [*arg* ...]
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **rdpol** *policy volume* [*plex*]
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **start** *volume* ...
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **startall**
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **stop** *volume* ...
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **stopall**
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **resync** *volume* ...
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **recover** *volume* [*subdisk*] ...
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **maint** *volume* ...
vxvol [**–Vf**] [**–g** *diskgroup*] [**–U** *usetype*] [**–o** *useopt*] **set** *attribute*=*value* ... [––]
volume ...

DESCRIPTION

The **vxvol** utility performs Volume Manager operations on volumes. The first operand is a keyword that determines the specific operation to perform. The remaining operands specify configuration records to which the operation is to be applied.

Each operation can be applied to only one disk group at a time, due to internal implementation constraints. Any *volume* operands will be used to determine a default disk group, according to the standard disk group selection rules described in **vxintro**(1M). A specific disk group can be selected with **–g** *diskgroup*.

The recognized operation keywords are:

init Perform an initialization action on a volume. This can be applied to volumes that were created by **vxmake** and that have not yet been initialized, or volumes that have been set to the uninitialized state with **vxmend fix empty**. The action to perform is specified by the *init_type* operand, which is usage-type-dependent. The *volume* operand determines which usage type to use for performing the operation.

rdpol Set the read policy for a volume based on the *policy* operand. These are the recognized read policies:

- **round** Use a round-robin read order among the enabled, readable plexes associated with the volume. No *plex* operand should be specified for the **round** read-policy type.
- **prefer** Read preferentially from the plex named by the *plex* operand. If the plex is enabled, readable, and associated with the volume, then any read operation on the volume results in a read from that plex if all blocks requested in the read are contained in the plex. The *plex* operand is required for the **prefer** read-policy type.
- **select** Select a default policy based on plex associations to the volume. For a volume that contains one enabled, striped plex, the default is to prefer that plex. For any other set of plex associations, the default is to use a round-robin policy. No *plex* operand should be specified for the **select** read-policy type.

start Enable disabled or detached volumes named by the *volume* operands. The process of enabling a volume is a highly usage-type-dependent operation and may result in transfers of data between plexes associated with the volume.

If the **start** operation is applied to an uninitialized volume (for example, a volume just created by **vxmake**), a default initialization will be used to initialize and enable the volume.

If the volume is not normally started because failures and disk removals have left all associated plexes with invalid data, the **–f** option can be used to try to start the volume, anyway. This can be used after replacing disks to enable the volume so that its contents can be restored from backup or reinitialized.

startall Attempt to start all volumes that are disabled. If a **–U** *usetype* option is specified, then attempt to start all disabled volumes with the indicated usage type. This operation will not start uninitialized volumes. By default, start all volumes in the **rootdg** disk group. A different disk group can be specified with the **–g** option.

stop Disable the enabled or detached volumes named by the *volume* operands.

The **stop** operation provides an interface to the usage type of a volume for shutting down operations on a volume in a clean manner. The specific method for cleanly stopping a volume, and the precise meaning of ''clean'' are both highly usage-type-dependent. By convention, **–f** can be used to force stopping of a volume that is in use, forcing I/O failures to be returned for any further volume device operations.

stopall Attempt to stop all volumes that are enabled. If a **–o** *usetype* option is specified, then attempt to stop all disabled volumes with the indicated usage type. By default, stop all volumes in the **rootdg** disk group. A different disk group can be specified with the **–g** option.

resync Examine all volumes named by the *volume* operands and perform any synchcronization operations that are required. The exact procedure for this operation is usage-type specific.

recover

Some usage types support the concept of recovery of data for objects within a volume different from the mirror resyncrhonization model. The **recover** operation is used to specify this type of recovery. The exact procedure for this operation is usage-type specific.

maint Detach each volume named by the *volume* operands and make the plex devices associated with those volumes accessible to regular system utilities. When a volume is detached, normal read and write operations to the volume fail, although most volume ioctl operations can still be used. Normal reads and writes can be used on the plex devices for the associated plexes. This operation could be used, for example, before using the **fsck** utility to decide which of several associated plexes should be used for reviving other plexes in a volume.

set Change specific volume characteristics. The changes to be made are given by arguments immediately after the **set** keyword of the form *attribute*=*value*. The set

of volumes affected by the operation are given after these operands; thus the attribute list ends with an operand that does not contain an equal sign. To allow for volume names that contain an equal sign, an operand of –– can be used to terminate the attribute list. Each usage-type represented by the list of volume operands is called once, with the set of all volumes with that usage type.

The set of *attribute*=*value* attribute arguments that are recognized depends upon the volume usage-type. However, an attribute argument of the form

len=*number*

is expected to be interpreted (if at all) as requesting a change in the length of a volume regardless of the volume's usage-type. The *number* value is interpreted as a standard length number (see **vxintro**(1M).

OPTIONS

The following options are recognized:

–g *diskgroup*
Specify the disk group for the operation. The disk group can be specified either by name or by disk group ID. See **vxdg**(1M) for more information on disk groups.

–U *usetype*
Force the operation to be performed by the usage-type utility for this usage type.

–V Write a list of utilities that would be called from **vxvol**, along with the arguments that would be passed. The **–V** option performs a ''mock run'' so the utilities are not actually called, and no changes are made to the volume configuration database.

–f Force an operation in some situations where the operation has questionable semantics. For example, **–f** may be used to reduce the length of a volume with **vxvol set**, to stop a volume that is currently open or mounted as a file system, or to attempt to start a volume that has no plexes with valid data.

–o *useopt*
Pass in usage-type-specific options to the operation. By convention, the following usage-type-specific options should be implemented by all usage types:

bg Perform any extended revive operations in background processes after the volume and one or more plexes have been enabled. A volume that is started or whose length is changed successfully with this option will be usable immediately after the operation completes, although recovery operations may affect performance of the volume for an extended period of time.

plexfork[**=***count*]
Perform up to the specified number of plex revive operations simultaneously. If no *count* is specified, then a suitable small number is used (normally 10).

delayrecover
Do not perform any plex revive operations when starting a volume.

Simply enable the volume and any plexes. This may leave some stale plexes, and may leave a mirrored volume in a special read-writeback (**NEEDSYNC**) recover state that performs limited plex recovery for each read to the volume.

slow[**=***iodelay*]
Reduce the system performance impact of plex recovery operations and volume length changes. Startup recovery and length change consistency operations are usually a set of short operations on small regions of the volume (normally from 16K bytes to 128K bytes). This option inserts a delay between the recovery of each such region. A specific delay can be specified with *iodelay* as a number of milliseconds, or else a default is chosen (normally 250 milliseconds).

iosize=*size*
Perform recovery operations in regions with the length specified by *size*, which is a standard Volume Manager length number (see **vxintro**(1M)). Specifying a larger number typically causes the operation to complete sooner, but with greater impact on other processes using the volume. The default I/O size is typically 32 kilobytes.

verbose
Print a message for each volume that is successfully started. Without this option, messages appear only for volumes that fail to start.

FSGEN AND GEN USAGE TYPES

The **fsgen** and **gen** usage types provide the same semantics for all operations of the **vxvol** utility. However, some options are provided only by the **gen** usage type.

In addition to the standard **–o** options required for all usage types, the **fsgen** and **gen** usage types provide the following additional options:

force Force an operation that is not normally performed as part of the operational model of the Volume Manager and may have adverse effects on data. This is the same as **–f**.

norecov
This can only be used with the **gen** usage type. Prevent the **start** operation from recovering plexes through the **vxplex** utility. Instead, all **STALE** and **ACTIVE** plexes are simply treated as equivalent to **CLEAN** plexes, and are thus enabled without being made consistent. This can be used for volumes whose contents are recreated for each use.

An example of a possible use for this attribute is a swap area and the **/tmp** file system. In the case of **/tmp**, the model assumes that **mkfs** is used to create an empty file system after the volume has been started.

Limitations and extensions for the **fsgen** and **gen** usage types consist of the following:

init These are the recognized uses of the **vxvol init** operation:

vxvol init clean *volume* [*plex*]
Set the state for the specified plex to **CLEAN**, and set all other plexes to

STALE. The **vxvol start** operation can then be used to recover the volume from the **CLEAN** plex. This operation requires that the volume not be enabled.

If the specified volume has only one plex, then the *plex* argument is not required as it defaults to that plex. If specified, then the *plex* argument must represent a plex that is associated with the volume.

vxvol init active *volume*
Set the state for all plexes associated with *volume* to **ACTIVE** and enable the volume and its plexes. This is used to initialize a single or multiple-plex volume where all plexes are known to have identical contents.

vxvol init enable *volume*
Enable the volume and its plexes but leave the volume uninitialized. This operation can be used only for non-enabled volumes. It is used to temporarily enable a volume so that data can be loaded onto it to make it consistent. Once the data has been loaded, **init active** should be used to fully enable the volume. **init active** could be used, for example, if a complete image of the volume is to be loaded from a tape.

vxvol init zero *volume*
Write zero blocks to all plexes in the volume, up to the length of the volume. After the writes complete, the state of each plex is set to **ACTIVE** and the volume and its plexes are enabled. **init zero** *volume* could be used, for example, before running **mkfs** to put a file system on the volume.

If this operation is interrupted by a signal, then an attempt is made to restore all affected records to their original state, or to a state that is roughly equivalent to their original state. If this attempt is interrupted, such as through another signal, then the user many need to perform some cleanup. A set of commands to perform this cleanup are written to the standard error before the **volume** utility exits.

start Starting an uninitialized **gen** or **fsgen** volume enables the volume and its plexes, sets the plexes to the **ACTIVE** state, and recovers the plexes to ensure that each plex has the same contents. If the volume has only one plex, then the volume is immediately set to the **ACTIVE** state; otherwise, the volume is set to the **SYNC** state and a special read / write-back mode is set to recover regions of the volume on every read operation. The volume is then read from beginning to end to make all plexes consistent, then the volume is set to the **ACTIVE** state.

Starting a volume with no active dirty region logging involves enabling all **CLEAN** and **ACTIVE** plexes and putting them in the **ACTIVE** state. If an I / O failure was logged against the plex, or if a disk replacement caused a plex to become stale, then the plex is considered **STALE**. If any of the subdisks for the plex reside on a removed or inaccessible disk, then the plex is ignored for the purposes of starting the volume.

If two or more plexes were enabled, and if the volume was active at the time the

system went down, then the state for the volume is set to **SYNC** and a special read / write-back recovery mode is used to recover consistency of the volume, segment-by-segment, on every read. A process (in the background with the **-o bg** option) to recover consistency for the entire length of the volume is then started.

If any plexes were considered **STALE**, then those plexes are attached by calling **vxplex att**. The number of concurrent plex attach operations are limited based on the rules for **–o plexfork**.

Recovery of plexes with a dirty region log uses the same rules as for volumes without a valid dirty region log, except that recovery of non-stale plexes is done by scanning the contents of the dirty region log and recovering consistency for those regions listed in the log as requiring recovery.

In addition to enabling the volume and managing the recovery of plex consistency, starting a volume clears any transient operations that were being applied to a volume before the system was rebooted. Starting a volume dissociates and removes temporary plexes or subdisks, and dissociates plexes that were being attached if the attach operation did not complete. Snapshot plexes created by **vxassist** are also removed.

If the volume is unstartable because there are no valid, non-stale plexes and the **–f** flag is then specified, all **STALE** plexes that do not contain unusable subdisks (subdisks on failed or removed disks) will be changed to **ACTIVE**. The volume will then be started and synchronized from those plexes.

stop Stopping an **fsgen** or **gen** volume disables the volume and its associated plexes. In addition, the utility state for each **ACTIVE** plex is changed as follows:

- If the plex is detached or disabled, set the state for the plex to **STALE**. If all plexes are set to **STALE**, then the volume cannot be started until **vxmend** is used to change the state of one or more plexes to **CLEAN** or **ACTIVE**. A plex normally becomes detached as a result of an I / O error on the plex, or a disk failure or replacement. I / O failures will not normally detach the last remaining enabled plex in a volume, so disk removal operations are the only normal operational method of making a volume unstartable.
- If the plex is volatile, i.e., one of the subdisks in the plex is defined on a disk with the volatile attribute (see **vxdisk**(1M)), then set the plex state to **STALE**.
- If the volume is enabled and the plex is also enabled, then set the plex state to **CLEAN**.
- If the volume is detached and the plex is enabled, then the plex state is left as **ACTIVE**. A volume can be left detached, with remaining valid plexes, only as a result of calling **vxvol maint** to detach an enabled volume.

Normally, the **stop** operation fails if any extended operations are using the volume or any of its associated plexes. Such operations are detected as a nonempty value for the **tutil0** field in a volume or plex record. If the **–f** option is specified, then the **stop** operation ignores volume and plex **tutil0** fields.

The **–f** option must also be given to force the stopping of a volume that is open or

mounted as a file system. In this case, a warning message is still written to the standard error, but the **stop** operation is not otherwise affected. Stopping an open or mounted volume is not normally advisable. Volumes that have possibly differing plex contents will be re-synchronized to contain consistent data. Any such volumes that are in the **NEEDSYNC** state will be recovered using a read / write-back recovery mode and then put into the **ACTIVE** state.

Plexes in the **SYNC** state may already be under recovery and the volume command will take no action to recover them unless the command was invoked with the **-o force** option.

maint The **–f** option is required to detach an enabled volume. Also, a warning is written to the standard error for volumes that are open or mounted.

set The attributes that can be changed are:

len=*number*

Change the length of each volume specified by the *volume* operands to be *number* sectors. The *number* is a standard Volume Manager length number (see **vxintro**(1M)). Decreasing the length of a volume requires **–f**.

If the volume is enabled, then count the number of enabled, read-write plexes that would remain complete after the length change. The operation fails if this number would become zero, but the number of sparse plexes would become greater than 1. Changing the length of a volume with one enabled plex beyond the length of the plex requires use of the **–f** option.

If the volume is not enabled, count the number of **CLEAN** and **ACTIVE** plexes that would remain complete after the length change, then use the algorithm mentioned above for determining whether the operation is allowed or requires use of **–f**.

In order to ensure that the new region of the volume is consistent across all plexes of the volume, the volume is put into a **SYNC** state and read / write-back mode, and a read loop is now performed against the volume. Once this loop has completed, the volume is put back into the **ACTIVE** state.

logtype=*type*

Set the type of logging to be used on the volume. This change can be applied only to volumes that are stopped and that have no *ACTIVE* plexes. Allowed log types are *drl* (logs the regions involved in all volume writes), *none* (never does logging), and *undef* (never does logging). If the logging type is set to *undef*, then a future *vxsd aslog* or *vxplex att* operation will change it to *drl*. See the *fsgen* and *gen* sections of *vxsd(8)* and *vxplex(8)* for more information.

loglen=*size*

Set the size for logs used with the volume. If the logging type is *drl*, then this value must be 2 to 10 sectors. The size value are standard volume

manager length numbers (see *vxintro(8)*).

startopts=*volume_options*
Set options that are applied to the volume every time the volume is started, independently of options specified with the **volume start** command. This is a set of comma-separated options of the same form used with the **–o** option letter. At the present time, only the **noattach** and **verbose** options can be applied to volumes in this manner. Unrecognized or inappropriate options are ignored.

RAID-5 USAGE TYPE

In addition to the stanard **–o** options required for all usage types, the **raid5** usage type provides the following options:

checkpt=*size*
Set the checkpoint size for a volume. A complete resynchronization of a volume via **VOL_R5_RESYNC** ioctls can take an extended amount of time. It is conciev-able in some circumstances that the operation could be stopped before it complete (such as by a system crash). To avoid having to restart the synchronization from the beginning of the volume, after a certain amount of the volume has been syncrhonized a transaction is issued to record the offset to which the resyncrhonization has completed. This size is called the *checkpoint length* and can be set using the **checkpt** option. The default checkpoint length is sixty-four megabytes.

force Force an operation that is not normally performed as part of the operational model of the Volume Manager. It may have adverse effects on the data contained by the volume. This is the same as **–f**.

delayrecover
This prevents the **start** operation from undergoing some recovery operations. RAID-5 logs, if any are valid, will still be replayed; however, no parity resychronizations or subdisk recoveries will be performed.

unsafe This allows access to certain volumes earlier in the starting process than is normally allowed by the operating process of RAID-5 volumes. This can have adverse effects on the data, and can also result in the RAID-5 volume becoming unusable following a system crash or a power failure.

syncstartok
This allows the **delayrecover** option to be ignored if the volume must undergo parity resynchronizations or subdisk recoveries before the volume can be enabled.

Limitations and extensions for the **raid5** usage type consist of the following:

init The **raid5** usage type recognizes the following uses of the **init** operation:

vxvol init active *volume*
Zero the RAID-5 logging plexes, if any, and make the volume available for use. The parity in the volume is marked as stale, though no parity resynchronization is performed; the volume is left with a state of *NEEDSYNC*.

vxvol init zero *volume*

Writes zeros to the RAID-5 logging plexes, if any, and write zeros to the entire length of the volume. This is achieved by issuing the **VOL_R5_ZERO** ioctl for the entire altitude of the volume. The volume is left in the *ACTIVE*.

start Starting an uninitialized volume (one with a state of **EMPTY**) will zero any RAID-5 log plexes and then resyncrhonize the parity of the volume by issuing **VOL_R5_RESYNC** ioctls. All subdisks are marked as non-stale and read-write. The volume and RAID-5 plex are then enabled and marked as **ACTIVE**, and all valid RAID-5 logging plexes are marked as **LOG**. If any RAID-5 logging plex proves to be invalid (such as having it's **NODAREC** flag set) their state is set to **BADLOG**.

Starting a volume that has been shut down cleanly or is not marked as dirty will enable the RAID-5 plex and RAID-5 logging plexes and set the volume kernel state detached to zero the RAID-5 logging plexes for the volume, if any. Once this is completed, all valid RAID-5 logging plexes are set to **LOG** and the volume is enabled and put in the **ACTIVE** state.

Starting a volume that was not shut down cleanly requires that the parity be resyncrhonized. If the volume has valid RAID-5 logging plexes, the volume is first detached and has it's state set to **REPLAY** and all logging plexes and the RAID-5 plex are enabled. If there are any valid RAID-5 logging plexes, their contents are read and the data it contains is written to the appropriate regions of the RAID-5 plex. If reading the RAID-5 logs fails, the logs are marked as invalid and the parity is resyncrhonized as if there were no logs (see below). Once the replay is complete, the RAID-5 logs are enabled and the volume is enabled and it's state is set to **ACTIVE**.

If the volume needs resyncrhonization and no valid logging plexes exist, the parity must be fully resynchronized. The volume is enabled and it's state is set to **RESYNC** and the RAID-5 plex is enabled. If usable RAID-5 plexes are available, but simply contained invalid data, they are zeroed. The parity is then resynchronized by issuing **VOL_R5_RESYNC** ioctls for the entire altitude of the volume. Once this is completed, the volume's state is set to **ACTIVE**. Any usable RAID-5 logs are enabled and set to the **LOG** state.

If a volume requires full resyncrhonization (i.e. has no usable logs) and the RAID-5 plex has stale or unusable subdisks, the volume is unusable and the **start** operation will fail. This can be overridden by using the **–f** flag or the **–o force** option . In this case, any stale subdisks are marked as non-stale and a full resyncrhonization is performed; however, this may result in some invalid data being introduced into the volume. If multiple subdisks at the same altitude in the RAID-5 plex are unusable (such as because they have their **NODEVICE** flag set), the volume is unusable and cannot be overridden.

Once any parity resyncrhonization has been completed, any subdisks still marked as stale are recovered. This is done by marking the subdisk as stale and

write-only and issuing **VOL_R5_RECOVER** ioctls to regenerate the data on the stale subdisks. The subdisk is then marked as non-stale and read-write.

If the **–o delayrecover** option is specified, the only recovery that will be performed are log replays. If the volume requires a parity resyncrhonization, it will be enabled and left in the **NEEDSYNC** operation and it's parity is marked as stale. Any subdisk recoveries that are needed will not be performed, and the stale subdisks are marked as stale.

Normally, if a volume has no RAID-5 logs, it will not be enabled with a stale subdisk or an unusable subdisk because were the system to crash or a power failure occured while the volume was is in use, the parity could become stale and the volume would be unusable. This behaviour can be overridden by specifying the **–o unsafe** option. This will cause the volume to be enabled during the above situations. As the name suggests, this is considered unsafe because doing so could cause a loss of data.

If only the **–o delayrecover** option is specified to start a volume with a stale subdisk or an unusable subdisk, the start operation will fail. In this case, the **delayrecover** option can be ignored by also specifying the **–o syncstartok** option.

stop Stopping a **raid5** volume disables the volume and it's associated plexes. If the volume is in the *SYNC* state, it is changed to the *NEEDSYNC* state so that recovery will be performed at the next start. Any invalid or detached RAID-5 logs are set to the *BADLOG* state so that they will not be used during the next start.

Normally, the **stop** operation will fail if any extended operations are using the volume or any of it's plexes. Such operations are detected as a non-empty value for the **tutil0** field in a volume or plex record. If the **–f** option is specified, then the **stop** operation ignores volume and plex **tutil0** fields.

resync The **resync** operation examines the named volumes to see if they are enabled and if the parity in any part of a volume is stale; this is normally indicated by a volume state of **NEEDSYNC**. If so, the volume is placed in the **SYNC** state and **VOL_R5_RESYNC** ioctls are issued to resyncrhonize the parity in those regions. Upon completion, the volume is placed in the **ACTIVE** state.

recover

The **raid5** usage type supports the following invocations of the **recover** operation:

recover [*subdisk ...]*

The **recover** operation can be used to initiate a recovery of subdisks containing invalid data. If subdisks are specified and are stale, the are recovered in the order specified. This is done by setting the stale and write-only flags on the subdisks and issuing **VOL_R5_RECOVER** ioctls to regenerate the data. After a successful recovery, the subdisk is marked as non-stale and read-write.

If no subdisk arguments are specified, the subdisks of the RAID-5 plex of the volume are checked to see if they are stale or have invalid contents. If any are found, they are recovered as described above.

set The attributes that can be set for **raid5** volumes are:

len=*number*

Change the length of the volumes specified to be *number* sectors. The *number* is a standard Volume Manager length specification (see **vxintro**(1M)). Decreasing the length of a volume requires **–f**.

The volume length cannot be increased such that the RAID-5 plex is sparse in respect to the new volume length; this would make the volume unusable.

In order to assre that the new region of the volume is consistent, the new region of the volume (from the old length to the new length) is filled with zeros by issuing **VOL_R5_ZERO** ioctls before the length is reset.

loglen=*size*

Set the size of the RAID-5 log for the volume. This cannot be set if the volume has no logs. If the length is being increased, the operation will not be allowed if it would cause any of the RAID-5 logging plexes to become sparse in respect to the new length.

startopts=*volume_options*

Set options that are applied to the volume every time the volume is started, independently of options specified with the **volume start** command. This is a set of comma-separated options of the same form used with the **–o** option letter. Unrecognized or inappropriate options are ignored.

FILES

/etc/vx/type/*usetype* / vxvol

The utility that performs **volume** operations for a particular volume usage type.

/dev/vx/dsk/*group***/***volume*

The device node that can be used for mounting a file system created on the volume named **volume** in the disk group named **group**. Volumes in group **rootdg** are also directly under the **/dev/vx/dsk** directory.

/dev/vx/rdsk/*group***/***volume*

The device node that can be used for issuing raw I/O requests and also for issuing ioctl requests to the volume named *volume* in disk group named *group*. Volumes in group **rootdg** are also directly under the **/dev/vx/rdsk** directory.

EXIT CODES

The **vxvol** utility exits with a nonzero status if the attempted operation fails. A nonzero exit code is not a complete indicator of the problems encountered, but rather denotes the first condition that prevented further execution of the utility.

See **vxintro**(1M) for a list of standard exit codes.

SEE ALSO **vxintro**(1M), **vxassist**(1M), **vxinfo**(1M), **vxmend**(1M), **vxplex**(1M), **vxrecover**(1M).

Index