

Veritas™ Storage Foundation Advanced Features Administrator's Guide

AIX

5.1 Service Pack 1

Veritas Storage Foundation™ Advanced Features Administrator's Guide

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

Product version: 5.1 SP1

Document version: 5.1SP1.1

Legal Notice

Copyright © 2010 Symantec Corporation. All rights reserved.

Symantec, the Symantec logo, Veritas, Veritas Storage Foundation, CommandCentral, NetBackup, Enterprise Vault, and LiveUpdate are trademarks or registered trademarks of Symantec corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Symantec Corporation and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. SYMANTEC CORPORATION SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, "Rights in Commercial Computer Software or Commercial Computer Software Documentation", as applicable, and any successor regulations. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043
<http://www.symantec.com>

Technical Support

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

Symantec's support offerings include the following:

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

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

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

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

Contacting Technical Support

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

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

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

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

- Product release level

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

Licensing and registration

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

www.symantec.com/business/support/

Customer service

Customer service information is available at the following URL:

www.symantec.com/business/support/

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

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

Support agreement resources

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

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

Documentation

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

docs@symantec.com

About Symantec Connect

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

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

Contents

Technical Support	4
Section 1 Storage Foundation advanced features	21
Chapter 1 Introducing Veritas Storage Foundation™ Advanced Features	23
About Storage Foundation management features	23
Storage management features for Storage Foundation products	26
Section 2 Improving performance with database accelerators	31
Chapter 2 Overview of database accelerators	33
About Storage Foundation database accelerators	33
About Quick I/O	34
How Quick I/O works	34
How Quick I/O improves database performance	35
About Oracle Disk Manager	35
About Cached ODM	36
Chapter 3 Improving DB2 performance with Veritas Quick I/O	39
How to set up Quick I/O	39
Creating database containers as Quick I/O files using qiomkfile for DB2	40
Preallocating space for Quick I/O files using the setext command	42
Accessing regular VxFS files as Quick I/O files	43
Converting DB2 containers to Quick I/O files	44
About sparse files	49
Displaying Quick I/O status and file attributes	50

	Extending a Quick I/O file in a DB2 environment	51
	Monitoring tablespace free space with DB2 and extending tablespace containers	53
	Recreating Quick I/O files after restoring a database	55
	Disabling Quick I/O	56
Chapter 4	Improving Sybase performance with Veritas Quick I/O	57
	How to set up Quick I/O	57
	Preallocating space for Quick I/O files using the setext command	58
	Creating database files as Quick I/O files with qiomkfile	59
	Accessing regular VxFS files as Quick I/O files	62
	Converting Sybase files to Quick I/O files	64
	Displaying Quick I/O status and file attributes	71
	Extending a Quick I/O file in a Sybase environment	72
	Recreating Quick I/O files after restoring a database	74
	Disabling Quick I/O	75
Chapter 5	Improving DB2 database performance with Veritas Cached Quick I/O	77
	Tasks for setting up Cached Quick I/O	77
	Enabling Cached Quick I/O on a file system	78
	Enabling and disabling the qio_cache_enable flag	78
	Making Cached Quick I/O settings persistent across reboots and mounts	79
	Using vxtunefs to obtain tuning information	80
	Determining candidates for Cached Quick I/O	81
	Collecting I/O statistics	82
	About I/O statistics	82
	Effects of read-aheads on I/O statistics	84
	Other tools for analysis	84
	Enabling and disabling Cached Quick I/O for individual files	84
	Setting cache advisories for individual files	85
	Making individual file settings for Cached Quick I/O persistent	85
	Determining individual file settings for Cached Quick I/O using qioadmin	86

Chapter 6	Improving Sybase database performance with Veritas Cached Quick I/O	89
	Tasks for setting up Cached Quick I/O	89
	Enabling Cached Quick I/O on a file system	90
	Enabling and disabling the qio_cache_enable flag	90
	Making Cached Quick I/O settings persistent across reboots and mounts	91
	Using vxtunefs to obtain tuning information	91
	Determining candidates for Cached Quick I/O	93
	Collecting I/O statistics	93
	About I/O statistics	94
	Effects of read-aheads on I/O statistics	95
	Other tools for analysis	96
	Enabling and disabling Cached Quick I/O for individual files	96
	Setting cache advisories for individual files	96
	Making individual file settings for Cached Quick I/O persistent	97
	Determining individual file settings for Cached Quick I/O using qioadmin	98
Chapter 7	Improving database performance with Veritas Concurrent I/O	101
	About Concurrent I/O	101
	How Concurrent I/O works	101
	Enabling and disabling Concurrent I/O	102
	Enabling Concurrent I/O for DB2	102
	Disabling Concurrent I/O for DB2	105
	Enabling Concurrent I/O for Sybase	105
	Disabling Concurrent I/O for Sybase	106
Section 3	Storage Foundation Thin Storage optimization	107
Chapter 8	About SF Thin Storage optimization solutions	109
	About SF solutions for thin optimization	109
	About Thin Storage	110
	About Thin Provisioning	110
	About SF Thin Reclamation feature	110
	About SmartMove	110
	Using SmartMove with Thin Provisioning	111

Chapter 9	Migrating data from thick storage to thin storage	113
	About using SmartMove to migrate to Thin Storage	113
	Setting up SmartMove	113
	Displaying the SmartMove configuration	114
	Changing the SmartMove configuration	114
	Migrating to thin provisioning	114
Chapter 10	Using SF Thin Reclamation	117
	Reclamation of storage on thin reclamation arrays	117
	Identifying thin and thin reclamation LUNs	117
	How reclamation on a deleted volume works	118
	Thin Reclamation of a disk, a disk group, or an enclosure	119
	Thin Reclamation of a file system	120
	Triggering space reclamation	121
	Monitoring Thin Reclamation using the vxtask command	121
Section 4	Making point-in-time copies	123
Chapter 11	Understanding point-in-time copy methods	125
	About point-in-time copies	125
	Implementing point-in time copy solutions on a primary host	126
	Implementing off-host point-in-time copy solutions	127
	About point-in-time copy technology	133
	Volume snapshots	134
	Disk group split/join	136
	Storage Checkpoints	137
	Veritas FlashSnap Agent for Symmetrix	138
	Point-in-time copy use cases	138
Chapter 12	Setting up volumes for instant snapshots	141
	About setting up volumes for instant snapshots	141
	Additional preparation activities	142
	Preparing a volume for instant snapshot operations	143
	Considerations for placing DCO plexes	147
	Creating a volume for use as a full-sized instant snapshot	150
	Creating a shared cache object	151
	Tuning the autogrow attributes	153

Chapter 13	Online database backup	155
	About online database backup	155
	Making a backup of an online database on the same host	156
	Making an off-host backup of an online database	160
Chapter 14	Off-host cluster file system backup	167
	About off-host cluster file system backup	167
	Mounting a file system for shared access	168
	Using off-host processing to back up cluster file systems	169
	Reattaching snapshot plexes	175
Chapter 15	Decision support	177
	About decision support	177
	Creating a replica database on the same host	178
	Creating an off-host replica database	184
	Resynchronizing the data with the primary host	191
	Reattaching snapshot plexes	192
Chapter 16	Database recovery	195
	About database recovery using Storage Checkpoints	195
	Creating Storage Checkpoints	196
	Rolling back a database	196
Chapter 17	Administering volume snapshots	199
	About volume snapshots	199
	Traditional third-mirror break-off snapshots	201
	Full-sized instant snapshots	202
	Space-optimized instant snapshots	204
	Emulation of third-mirror break-off snapshots	205
	Linked break-off snapshot volumes	205
	Cascaded snapshots	207
	Creating a snapshot of a snapshot	208
	Creating multiple snapshots	210
	Restoring the original volume from a snapshot	211
	Creating instant snapshots	212
	Preparing to create instant and break-off snapshots	214
	Creating and managing space-optimized instant snapshots	218
	Creating and managing full-sized instant snapshots	221
	Creating and managing third-mirror break-off snapshots	223
	Creating and managing linked break-off snapshot volumes	226

Creating multiple instant snapshots	228	
Creating instant snapshots of volume sets	229	
Adding snapshot mirrors to a volume	231	
Removing a snapshot mirror	232	
Removing a linked break-off snapshot volume	232	
Adding a snapshot to a cascaded snapshot hierarchy	232	
Refreshing an instant snapshot	233	
Reattaching an instant snapshot	233	
Reattaching a linked break-off snapshot volume	234	
Restoring a volume from an instant snapshot	235	
Dissociating an instant snapshot	236	
Removing an instant snapshot	236	
Splitting an instant snapshot hierarchy	237	
Displaying instant snapshot information	237	
Controlling instant snapshot synchronization	239	
Listing the snapshots created on a cache	241	
Tuning the autogrow attributes of a cache	241	
Monitoring and displaying cache usage	242	
Growing and shrinking a cache	243	
Removing a cache	244	
Creating traditional third-mirror break-off snapshots	244	
Converting a plex into a snapshot plex	248	
Creating multiple snapshots with the vxassist command	249	
Reattaching a snapshot volume	250	
Adding plexes to a snapshot volume	251	
Dissociating a snapshot volume	252	
Displaying snapshot information	252	
Adding a version 0 DCO and DCO volume	253	
Specifying storage for version 0 DCO plexes	255	
Removing a version 0 DCO and DCO volume	257	
Reattaching a version 0 DCO and DCO volume	257	
Chapter 18	Administering snapshot file systems	259
	About snapshot file systems	259
	How a snapshot file system works	260
	Snapshot file system backups	261
	Snapshot file system performance	262
	About snapshot file system disk structure	262
	Differences between snapshots and Storage Checkpoints	263
	Creating a snapshot file system	264
	Backup examples	265

Chapter 19	Administering Storage Checkpoints	267
	About Storage Checkpoints	267
	Distinguishing between Storage Checkpoints and snapshots	268
	Operation of a Storage Checkpoint	269
	Copy-on-write	271
	Types of Storage Checkpoints	272
	Data Storage Checkpoints	272
	Nodata Storage Checkpoints	273
	Removable Storage Checkpoints	274
	Non-mountable Storage Checkpoints	274
	Storage Checkpoint administration	274
	Creating a Storage Checkpoint	275
	Removing a Storage Checkpoint	275
	Accessing a Storage Checkpoint	276
	Converting a data Storage Checkpoint to a nodata Storage Checkpoint	278
	Storage Checkpoint space management considerations	286
	Restoring from a Storage Checkpoint	287
	Restoring a file from a Storage Checkpoint	287
	Storage Checkpoint quotas	292
Chapter 20	Administering FileSnaps	293
	About FileSnaps	293
	Properties of FileSnaps	294
	Creating FileSnaps	295
	Concurrent I/O to FileSnaps	295
	Copy-on-write and FileSnaps	295
	Reading from FileSnaps	296
	Block map fragmentation and FileSnaps	296
	Backup and FileSnaps	296
	Using FileSnaps	297
	Best practices with FileSnaps	298
	Virtual desktops	298
	Write intensive applications	298
	Data mining, reporting, and testing	299
	Comparison of the logical size output of the fsadm -S shared, du, and df commands	299

Chapter 21	Backing up and restoring with Netbackup in an SFHA environment	301
	About Veritas NetBackup	301
	About using Veritas NetBackup for backup and restore for DB2	302
	Components of Veritas NetBackup	302
	About performing a backup	303
	About configuring Veritas NetBackup for a DB2 EEE (DPF) environment	303
	About using NetBackup for backup and restore for Sybase	303
	About using Veritas NetBackup to backup and restore Quick I/O files for DB2	304
	About using Veritas NetBackup to backup and restore Quick I/O files for Sybase	305
Section 5	Maximizing storage utilization	307
Chapter 22	Understanding storage tiering with SmartTier	309
	About SmartTier	309
	SmartTier building blocks	310
	About VxFS multi-volume file systems	310
	About VxVM volume sets	311
	About volume tags	311
	How SmartTier works	312
	Moving files	313
	Moving sub-file objects	313
	SmartTier in a High Availability (HA) environment	314
Chapter 23	Creating and administering volume sets	315
	About volume sets	315
	Creating a volume set	316
	Adding a volume to a volume set	316
	Removing a volume from a volume set	317
	Listing details of volume sets	317
	Stopping and starting volume sets	318
	Raw device node access to component volumes	319
	Enabling raw device access when creating a volume set	320
	Displaying the raw device access settings for a volume set	321
	Controlling raw device access for an existing volume set	321

Chapter 24	Multi-volume file systems	323
	About multi-volume support	323
	About volume types	324
	Features implemented using multi-volume support	324
	Volume availability	325
	Creating multi-volume file systems	326
	Example of creating a multi-volume file system	326
	Converting a single volume file system to a multi-volume file system	327
	Adding a volume to and removing a volume from a multi-volume file system	329
	Adding a volume to a multi-volume file system	329
	Removing a volume from a multi-volume file system	329
	Forcibly removing a volume	329
	Moving volume 0	330
	Volume encapsulation	330
	Encapsulating a volume	330
	Deencapsulating a volume	331
	Reporting file extents	332
	Examples of reporting file extents	332
	Load balancing	333
	Defining and assigning a load balancing allocation policy	334
	Rebalancing extents	334
	Converting a multi-volume file system to a single volume file system	335
	Converting to a single volume file system	335
Chapter 25	Administering SmartTier	337
	About SmartTier	337
	Supported SmartTier document type definitions	339
	Placement classes	339
	Tagging volumes as placement classes	340
	Listing placement classes	340
	Administering placement policies	341
	Assigning a placement policy	341
	Unassigning a placement policy	341
	Analyzing the space impact of enforcing a placement policy	342
	Querying which files will be affected by enforcing a placement policy	342
	Enforcing a placement policy	342
	Validating a placement policy	344

File placement policy grammar	344
File placement policy rules	345
SELECT statement	345
CREATE statement	349
RELOCATE statement	351
DELETE statement	365
Calculating I/O temperature and access temperature	367
Multiple criteria in file placement policy rule statements	371
Multiple file selection criteria in SELECT statement clauses	371
Multiple placement classes in <ON> clauses of CREATE statements and in <TO> clauses of RELOCATE statements	372
Multiple placement classes in <FROM> clauses of RELOCATE and DELETE statements	373
Multiple conditions in <WHEN> clauses of RELOCATE and DELETE statements	373
File placement policy rule and statement ordering	374
File placement policies and extending files	376
Using SmartTier with solid state disks	376
Fine grain temperatures	377
Prefer mechanism	377
Average I/O activity	378
Frequent scans	378
Quick identification of cold files	379
Example placement policy when using solid state disks	380
 Section 6	
Migrating data	385
 Chapter 26	
Understanding data migration	387
Types of data migration	387
 Chapter 27	
Offline data migration	389
About VxVM and LVM	389
About Veritas Volume Manager	389
VxVM and LVM, a conceptual comparison	392
Coexistence of VxVM and LVM disks	395
Converting LVM, JFS and JFS2 to VxVM and VxFS	395
About converting LVM, JFS and JFS2 configurations	395
Initializing unused LVM physical volumes to VxVM disks	396
Converting LVM volume groups to VxVM disk groups	397
Restoring the LVM volume group configuration	409

	Examples of using vxconvert	409
	General information regarding conversion speed	413
	Estimating system down time	414
	About test cases	414
	Command differences	419
	Command differences between LVM and VxVM	419
	LVM and VxVM command equivalents	420
	Comparison of LVM and VxVM tasks	424
	Tasks with no direct LVM equivalents	434
	LVM features not supported in VxVM	436
	System Management Interface Tool (SMIT)	437
	About the AIX System Management Interface Tool	437
	Launching SMIT	437
	Administering disk groups in SMIT	438
	Administering disk devices in SMIT	439
	Administering volumes in SMIT	440
	Administering VxVM tunables in SMIT	441
	Administering DMP tunables in SMIT	442
Chapter 28	Online migration of a native file system to the VxFS file system	445
	About online migration	445
	Administrative interface for online migration	446
	Migrating a native file system to the VxFS file system	447
	Backing out an online migration operation	449
	VxFS features not available during online migration	449
Chapter 29	Migrating data between platforms	451
	Overview of CDS	451
	Shared data across platforms	452
	Disk drive sector size	453
	Block size issues	453
	Operating system data	453
	CDS disk format and disk groups	453
	CDS disk access and format	454
	Non-CDS disk groups	457
	Disk group alignment	457
	Setting up your system	459
	Creating CDS disks from uninitialized disks	460
	Creating CDS disks from initialized VxVM disks	461
	Creating CDS disk groups	462
	Converting non-CDS disks to CDS disks	463

	Converting a non-CDS disk group to a CDS disk group	464
	Verifying licensing	466
	Defaults files	467
	Maintaining your system	468
	Disk tasks	469
	Disk group tasks	471
	Displaying information	477
	Default activation mode of shared disk groups	480
	Additional considerations when importing CDS disk groups	480
	File system considerations	481
	Considerations about data in the file system	482
	File system migration	482
	Specifying the migration target	483
	Using the fscdsadm command	484
	Migrating a file system one time	487
	Migrating a file system on an ongoing basis	487
	When to convert a file system	489
	Converting the byte order of a file system	489
	Alignment value and block size	493
	Migrating a snapshot volume	493
Section 7	Reference	497
Appendix A	Recovering from CDS errors	499
	CDS error codes and recovery actions	499
Appendix B	Conversion error messages	503
	List of conversion error messages	503
Appendix C	Files and scripts for sample scenarios	507
	About files and scripts for sample scenarios	507
	Script to initiate online off-host backup of an Oracle database	509
	Script to put an Oracle database into hot backup mode	511
	Script to quiesce a Sybase ASE database	512
	Script to suspend I/O for a DB2 database	512
	Script to end Oracle database hot backup mode	513
	Script to release a Sybase ASE database from quiesce mode	513
	Script to resume I/O for a DB2 database	514
	Script to perform off-host backup	514
	Script to create an off-host replica Oracle database	515
	Script to complete, recover and start a replica Oracle database	517

	Script to start a replica Sybase ASE database	519
Appendix D	Preparing a replica Oracle database	521
	About preparing a replica Oracle database	521
	Text control file for original production database	524
	SQL script to create a control file	526
	Initialization file for original production database	526
	Initialization file for replica Oracle database	528
Glossary	531
Index	541

Section

1

Storage Foundation advanced features

- [Chapter 1. Introducing Veritas Storage Foundation™ Advanced Features](#)

Introducing Veritas Storage Foundation™ Advanced Features

This chapter includes the following topics:

- [About Storage Foundation management features](#)
- [Storage management features for Storage Foundation products](#)

About Storage Foundation management features

This guide documents the advanced features of Veritas Storage Foundation and High Availability products. It is a supplemental guide to be used in conjunction with Veritas Storage Foundation product guides.

Table 1-1 Veritas Storage Foundation management features

Feature	Uses
<p>Enhanced I/O methods enable you to improve database performance:</p> <ul style="list-style-type: none"> ■ Veritas Extension for Oracle Disk Manager See the <i>Veritas Storage Foundation: Storage and Availability Management for Oracle Databases</i> ■ Veritas Extension for Cached Oracle Disk Manager See the <i>Veritas Storage Foundation: Storage and Availability Management for Oracle Databases</i> ■ Veritas Quick I/O ■ Veritas Cached Quick I/O ■ Veritas Concurrent I/O 	<ul style="list-style-type: none"> ■ To improve Oracle performance and manage system bandwidth through an improved Application Programming Interface (API) that contains advanced kernel support for file I/O, use Veritas Oracle Disk Manager (ODM). ■ To enable selected I/O to use caching to improve ODM I/O performance, use Veritas Extension for Cached Oracle Disk Manager (Cached ODM). ■ To achieve raw device performance for databases run on Veritas File System file systems while providing the administrative advantages of using file systems, use Veritas Quick I/O. ■ To further enhance database performance by leveraging large system memory to selectively buffer the frequently accessed data, use Veritas Cached Quick I/O. ■ To achieve improved performance for DB2 or Sybase databases run on Veritas File System file systems, without restrictions on increasing file size, use Veritas Concurrent I/O.
<p>Thin Provisioning supports the use of Thin Storage for your data:</p> <ul style="list-style-type: none"> ■ Thin Provisioning ■ Thin Reclamation ■ SmartMove 	<ul style="list-style-type: none"> ■ To optimize the use of Thin Storage on new volumes, initialize the volumes using Thin Provisioning. ■ To optimize storage administration by reclaiming space deleted or shrunk on the Veritas Volume Manager volume or the Veritas File System file system, use Thin Reclamation. ■ To optimize the use of Thin Storage for Veritas Volume Manager volumes on Veritas File System , use SmartMove to enable copy operations to move only used blocks.

Table 1-1 Veritas Storage Foundation management features (*continued*)

Feature	Uses
<p>Point-in-time copy features enable you to capture an instantaneous image of actively changing data:</p> <ul style="list-style-type: none"> ■ FlashSnap ■ Database FlashSnap See the <i>Veritas Storage Foundation: Storage and Availability Management for Oracle Databases</i> ■ Storage Checkpoints ■ Database Storage Checkpoints See the <i>Veritas Storage Foundation: Storage and Availability Management for Oracle Databases</i> 	<ul style="list-style-type: none"> ■ To make a snapshot of a volume, use the Veritas Volume Manager Flashsnap feature. ■ To perform backup or other maintenance tasks while providing continuous data availability, use the Veritas Volume Manager Flashsnap feature. ■ To perform off-host processing while providing continuous data availability, use the Veritas Volume Manager Flashsnap feature. ■ To resynchronize an online point-in-time image of a database in an Oracle environment, use the Storage Foundation Database Flashsnap feature. ■ To reverse resynchronize an online point-in-time image of a database in an Oracle environment, use the Storage Foundation Database Flashsnap feature. ■ To create a mountable point-in-time image of a file system, use the Veritas File System Storage Checkpoints feature. ■ To create a mountable point-in-time image of a file system in an Oracle environment, use the Veritas File System Storage Checkpoints feature. ■ To use point-in-time copies to make a STANDBY database for troubleshooting, reporting, and quality assurance for databases, use the cloning commands for Database Storage Checkpoints or Database FlashSnap.
<p>SmartTier enables you to manage the growth of your database storage more efficiently.</p> <p>SmartTier for Oracle enables you to manage the growth of your Oracle database more efficiently</p> <p>See the <i>Veritas Storage Foundation: Storage and Availability Management for Oracle Databases</i></p>	<ul style="list-style-type: none"> ■ To monitor your file access patterns and manage the movement of files across storage tiers to optimize utilization, use SmartTier at the file level. ■ To monitor file access patterns for your Oracle database and manage the movement of files across storage tiers to optimize storage utilization, use SmartTier at the file level. ■ To monitor your access patterns for file objects in your Oracle database and manage the movement of database objects at the sub-file level between storage tiers to optimize storage utilization, use SmartTier at the sub-file level. ■ To monitor file access patterns for your Oracle database and manage the movement of database objects across storage tiers to optimize storage utilization, use SmartTier at the file level.

Table 1-1 Veritas Storage Foundation management features (*continued*)

Feature	Uses
<p>Data sharing options to enable you to migrate data:</p> <ul style="list-style-type: none"> ■ Offline migration tools ■ Online migration tools ■ Portable Data Containers (PDCs) 	<ul style="list-style-type: none"> ■ To migrate from AIX Logical Volume Manager to Veritas Volume Manager, use the Veritas Volume Manager utilities for offline migration. ■ To migrate from AIX Logical Volume Manager to Veritas Volume Manager with minimal downtime, use the Veritas Volume Manager utilities for online migration. ■ To migrate data between systems that are running the different operating system, use Portable Data Containers, also known as Cross-Platform Data Sharing (CDS).
<p>Symantec plug-ins for third-party applications enable you to view storage and cluster properties for your configuration inside third-party applications.</p> <ul style="list-style-type: none"> ■ Symantec Storage Plug-in for Oracle Enterprise Manager (OEM) ■ Symantec High Availability Plug-in for Oracle Enterprise Manager (OEM) ■ The Symantec Storage Plug-in for VMWare VCenter. 	<ul style="list-style-type: none"> ■ To view Veritas File System properties, Veritas Volume Manager volume and LUN information for database objects such as tablespace, redo logs, controlfile, datafiles and others through the Oracle Enterprise Manager (OEM) interface, use the Symantec Storage Plug-in for Oracle Enterprise Manager (OEM). ■ To view properties for hosts running Veritas Cluster Server (VCS) through the Oracle Enterprise Manager (OEM) interface, use the Symantec High Availability Plug-in for Oracle Enterprise Manager (OEM). ■ To view Veritas Cluster Server properties, Veritas File System properties, Veritas Volume Manager volume and LUN information through the VMWare VCenter interface, use the Symantec Storage Plug-in for VMWare VCenter.

Storage management features for Storage Foundation products

The following table lists the Veritas Storage Foundation product suites in which each advanced feature is available.

Table 1-2 Advanced features in Storage Foundation

Storage Foundation feature	Product licenses which enable this feature
Veritas Extension for Oracle Disk Manager	Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
Veritas Extension for Cached Oracle Disk Manager	Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA
Quick I/O	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System HA Storage Foundation Cluster File System Storage Foundation for Oracle RAC
Cached Quick I/O	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC

Table 1-2 Advanced features in Storage Foundation (*continued*)

Storage Foundation feature	Product licenses which enable this feature
Concurrent I/O	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA
Thin Reclamation	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
SmartMove	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
FlashSnap	Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC

Table 1-2 Advanced features in Storage Foundation (*continued*)

Storage Foundation feature	Product licenses which enable this feature
Storage Checkpoints	Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
SmartTier	Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
Portable Data Containers	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA Storage Foundation for Oracle RAC
Symantec Storage Plug-in for VMWare vCenter	Storage Foundation Basic Storage Foundation Standard Storage Foundation Standard HA Storage Foundation Enterprise Storage Foundation Enterprise HA Storage Foundation Cluster File System Storage Foundation Cluster File System HA

Note: SmartTier is an expanded and renamed version of the feature previously known as Dynamic Storage Tiering (DST).

Improving performance with database accelerators

- [Chapter 2. Overview of database accelerators](#)
- [Chapter 3. Improving DB2 performance with Veritas Quick I/O](#)
- [Chapter 4. Improving Sybase performance with Veritas Quick I/O](#)
- [Chapter 5. Improving DB2 database performance with Veritas Cached Quick I/O](#)
- [Chapter 6. Improving Sybase database performance with Veritas Cached Quick I/O](#)
- [Chapter 7. Improving database performance with Veritas Concurrent I/O](#)

Overview of database accelerators

This chapter includes the following topics:

- [About Storage Foundation database accelerators](#)
- [About Quick I/O](#)
- [About Oracle Disk Manager](#)
- [About Cached ODM](#)

About Storage Foundation database accelerators

The major concern in any environment is maintaining respectable performance or meeting performance SLAs. Veritas Storage Foundation improves the overall performance of database environments in a variety of ways.

- Quick I/O (QIO) and cached Quick I/O (CQIO) optimized for all database environments
- Concurrent I/O (CIO) optimized for DB2 and Sybase environments
- Oracle Disk Manager (ODM) and Cached Oracle Disk Manager (CODM) optimized specifically for Oracle environments

These database accelerator technologies enable database performance equal to raw disk partitions, but with the manageability benefits of a file system. With the Dynamic Multi-pathing feature of Storage Foundation, performance is maximized by load-balancing I/O activity across all available paths from server to array. Dynamic Multi-pathing supports all major hardware RAID vendors, hence there is no need for third-party multi-pathing software, reducing the total cost of ownership.

Storage Foundation database accelerators enable you to manage performance for your database with more precision.

- To achieve raw device performance for databases run on Veritas File System file systems, use Veritas Quick I/O.
- To further enhance database performance by leveraging large system memory to selectively buffer the frequently accessed data, use Veritas Cached Quick I/O.
- To achieve improved performance for DB2 or Sybase databases run on Veritas File System file systems, without restrictions on increasing file size, use Veritas Concurrent I/O.
- To improve Oracle performance and manage system bandwidth through an improved Application Programming Interface (API) that contains advanced kernel support for file I/O, use Veritas Oracle Disk Manager (ODM).
- To enable selected I/O to use caching to improve ODM I/O performance, use Veritas Extension for Cached Oracle Disk Manager (Cached ODM).

About Quick I/O

Veritas Quick I/O is a VxFS feature included in Veritas Storage Foundation Standard and Enterprise products that lets applications access preallocated VxFS files as raw character devices. Quick I/O provides the administrative benefits of running databases on file systems without the typically associated degradation in performance.

Using Quick I/O is recommended on DB2 databases prior to DB2 8.1 with FixPak 4. With DB2 8.1 with FixPak 4 or later, you can use the Veritas Concurrent I/O feature, which provides high-speed access for SMS tablespaces as well as DMS tablespaces.

Note: Symantec recommends that you use Veritas Extension for Oracle Disk Manager for Oracle.

See the Storage Foundation: Storage and Availability Management for Oracle guide.

How Quick I/O works

Veritas Quick I/O supports direct I/O and AIX Fastpath asynchronous I/O and enables databases to access regular files on a VxFS file system as raw character devices.

The benefits of using Quick I/O are:

- Improved performance and processing throughput by having Quick I/O files act as raw devices.
- Ability to manage Quick I/O files as regular files, which simplifies administrative tasks such as allocating, moving, copying, resizing, and backing up Sybase dataservers.
- Ability to manage Quick I/O files as regular files, which simplifies administrative tasks such as allocating, moving, copying, resizing, and backing up DB2 containers.

How Quick I/O improves database performance

Quick I/O's ability to access regular files as raw devices improves database performance by:

- Supporting AIX Fastpath asynchronous I/O
- Supporting direct I/O
- Avoiding kernel write locks on database files
- Avoiding double buffering

About Oracle Disk Manager

Veritas Extension for Oracle Disk Manager enhances file management and disk I/O throughput. The features of Oracle Disk Manager are optimized for Oracle 10g or later databases in a Veritas File System environment. Oracle Disk Manager enables you to improve database throughput for I/O intensive workloads with special I/O optimization.

Veritas Extension for Oracle Disk Manager supports Oracle Resilvering. With Oracle Resilvering, the storage layer receives information from the Oracle database as to which regions or blocks of a mirrored datafile to resync after a system crash. Oracle Resilvering avoids overhead from the Volume Manager Dirty Region Logging (DRL), which increases performance.

Oracle Disk Manager reduces administrative overhead by providing enhanced support for Oracle Managed Files. Veritas Extension for Oracle Disk Manager has Quick I/O-like capabilities, but is transparent to the user. Unlike Veritas Quick I/O, files managed using Veritas Extension for Oracle Disk Manager do not require special file naming conventions. The Oracle Disk Manager interface uses regular database files. If you are upgrading to Oracle 10g or later, you should convert from Quick I/O to Oracle Disk Manager.

Database administrators can choose the datafile type used with the Oracle product. Historically, choosing between file system files and raw devices was based on manageability and performance. The exception to this is a database intended for use with Oracle Parallel Server, which requires raw devices on most platforms. If performance is not as important as administrative ease, file system files are typically the preferred file type. However, while an application may not have substantial I/O requirements when it is first implemented, I/O requirements may change. If an application becomes dependent upon I/O throughput, converting datafiles from file system to raw devices is often necessary.

Oracle Disk Manager was designed to work with Oracle10g or later to provide both performance and manageability. Oracle Disk Manager provides support for Oracle's file management and I/O calls for database storage on VxFS file systems and on raw volumes or partitions. This feature is provided as a dynamically-loaded shared library with which Oracle binds when it is loaded. The Oracle Disk Manager library works with an Oracle Disk Manager driver that is loaded in the kernel to perform its functions.

If you are upgrading to Oracle10g or later, you should convert from Quick I/O to Oracle Disk Manager.

The benefits of using Oracle Disk Manager are as follows:

- True kernel asynchronous I/O for files and raw devices
- Reduced system call overhead
- Improved file system layout by preallocating contiguous files on a VxFS file system
- Performance on file system files that is equivalent to raw devices
- Transparent to users

For more information on using Oracle Disk Manager:

See *Veritas Storage Foundation: Storage and Availability Management for Oracle Databases*.

About Cached ODM

ODM I/O normally bypasses the file system cache and directly reads from and writes to disk. Cached ODM enables some I/O to use caching and read ahead, which can improve ODM I/O performance. Cached ODM performs a conditional form of caching that is based on per-I/O hints from Oracle. The hints indicate what Oracle will do with the data. ODM uses these hints to perform caching and read ahead for some reads, but ODM avoids caching other reads, even for the same file.

For more information on using Cached ODM:

See *Veritas Storage Foundation: Storage and Availability Management for Oracle Databases*.

Improving DB2 performance with Veritas Quick I/O

This chapter includes the following topics:

- [How to set up Quick I/O](#)
- [Creating database containers as Quick I/O files using qiomkfile for DB2](#)
- [Preallocating space for Quick I/O files using the setext command](#)
- [Accessing regular VxFS files as Quick I/O files](#)
- [Converting DB2 containers to Quick I/O files](#)
- [About sparse files](#)
- [Displaying Quick I/O status and file attributes](#)
- [Extending a Quick I/O file in a DB2 environment](#)
- [Monitoring tablespace free space with DB2 and extending tablespace containers](#)
- [Recreating Quick I/O files after restoring a database](#)
- [Disabling Quick I/O](#)

How to set up Quick I/O

Quick I/O is included in the VxFS package shipped with Veritas Storage Foundation Standard and Enterprise products. By default, Quick I/O is enabled when you mount a VxFS file system.

If Quick I/O is not available in the kernel, or a Veritas Storage Foundation Standard or Enterprise product license is not installed, a file system mounts without Quick I/O by default, the Quick I/O file name is treated as a regular file, and no error message is displayed. If, however, you specify the `-o qio` option, the `mount` command prints the following error message and terminates without mounting the file system.

```
VxFDD: You don't have a license to run this program
vxfs mount: Quick I/O not available
```

Depending on whether you are creating a new database or are converting an existing database to use Quick I/O, you have the following options:

If you are creating a new DB2 database to use Quick I/O:

- You can use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
- You can use the `setext` command to preallocate space for database files and create the Quick I/O files.

If you are converting an existing database:

- You can create symbolic links for existing VxFS files, and use these symbolic links to access the files as Quick I/O files.
See [“Accessing regular VxFS files as Quick I/O files”](#) on page 43.
- You can convert your existing DB2 database files to use the Quick I/O interface using the `qio_getdbfiles` and `qio_convertdbfiles` commands.
See [“Converting DB2 containers to Quick I/O files”](#) on page 44.

Creating database containers as Quick I/O files using `qiomkfile` for DB2

You can create Database Managed Space (DMS) containers with the type 'DEVICE' using Quick I/O. The best way to preallocate space for tablespace containers and to make them accessible using the Quick I/O interface is to use the `qiomkfile`. You can use the `qiomkfile` to create the Quick I/O files for either temporary or permanent tablespaces.

- Prerequisites
- You can create Quick I/O files only on VxFS file systems.
 - If you are creating containers on an existing file system, run `fsadm` (or similar utility) to report and eliminate fragmentation.
 - You must have read/write permissions on the directory in which you intend to create DB2 Quick I/O files.

- Usage notes
- The `qiomkfile` command creates two files: a regular file with preallocated, contiguous space, and a file that is a symbolic link pointing to the Quick I/O name extension.
 - See the `qiomkfile(1M)` manual page for more information.
- `-a` Creates a symbolic link with an absolute path name for a specified file. Use the `-a` option when absolute path names are required. However, the default is to create a symbolic link with a relative path name.
- `-e` Extends a file by a specified amount to allow DB2 tablespace resizing. See “[Extending a Quick I/O file in a DB2 environment](#)” on page 51.
- `-r` Increases the file to a specified size to allow DB2 tablespace resizing. See “[Extending a Quick I/O file in a DB2 environment](#)” on page 51.
- `-s` Specifies the space to preallocate for a file in bytes, kilobytes, megabytes, gigabytes, or sectors (512 bytes) by adding a `k`, `K`, `m`, `M`, `g`, `G`, `s`, or `S` suffix. The default is bytes—you do not need to attach a suffix to specify the value in bytes. The size of the file that is preallocated is the total size of the file (including the header) rounded to the nearest multiple of the file system block size.

Warning: Exercise caution when using absolute path names. Extra steps may be required during database backup and restore procedures to preserve symbolic links. If you restore files to directories different from the original paths, you must change the symbolic links that use absolute path names to point to the new path names before the database is restarted.

To create a container as a Quick I/O file using `qiomkfile`

- 1 Create a Quick I/O-capable file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -s file_size /mount_point/filename
```

- 2 Create tablespace containers using this file with the following SQL statements:

```
$ db2 connect to database
$ db2 create tablespace tbsname managed by database using \
( DEVICE /mount_point/filename size )
$ db2 terminate
```

An example to show how to create a 100MB Quick I/O-capable file named `dbfile` on the VxFS file system `/db01` using a relative path name:

```
$ /opt/VRTS/bin/qiomkfile -s 100m /db01/dbfile
$ ls -al
-rw-r--r--  1 db2inst1  db2iadml 104857600  Oct 2 13:42  .dbfile
lrwxrwxrwx  1 db2inst1  db2iadml      19  Oct 2 13:42  dbfile -> \
.dbfile::cdev:vxfs:
```

In the example, `qiomkfile` creates a regular file named `/db01/dbfile`, which has the real space allocated. Then, `qiomkfile` creates a symbolic link named `/db01/.dbfile`. This symbolic link is a relative link to the Quick I/O interface for `/db01/.dbfile`, that is, to the `.dbfile::cdev:vxfs:` file. The symbolic link allows `.dbfile` to be accessed by any database or application using its Quick I/O interface.

We can then add the file to the DB2 database `PROD`:

```
$ db2 connect to PROD
$ db2 create tablespace NEWTBS managed by database using \
( DEVICE '/db01/dbfile' 100m )
$ db2 terminate
```

Preallocating space for Quick I/O files using the `setext` command

As an alternative to using the `qiomkfile` command, you can also use the VxFS `setext` command to preallocate space for database files.

Before preallocating space with `setext`, make sure the following conditions have been met:

- | | |
|---------------|---|
| Prerequisites | ■ The <code>setext</code> command requires superuser (<code>root</code>) privileges. |
| Usage notes | ■ You can use the <code>chown</code> command to change the owner and group permissions on the file after you create it.
See the <code>setext (1M)</code> manual page for more information. |

To create a Quick I/O database file using `setext`

1 Access the VxFS mount point and create a file:

```
# cd /mount_point
# touch .filename
```

2 Use the `setext` command to preallocate space for the file:

```
# /opt/VRTS/bin/setext -r size -f noreserve -f chgsize \
.filename
```

3 Create a symbolic link to allow databases or applications access to the file using its Quick I/O interface:

```
# ln -s .filename::
```

4 Change the owner and group permissions on the file.

```
# chown user:group .filename
# chmod 660 .filename
```

An example to show how to access the mount point for DB2 /db01, create a container, preallocate the space, and change the permissions:

```
# cd /db01
# touch .dbfile
# /opt/VRTS/bin/setext -r 100M -f noreserve -f chgsize .dbfile
# ln -s .dbfile::
```

Accessing regular VxFS files as Quick I/O files

You can access regular VxFS files as Quick I/O files using the `::cdev:vxfs: name` extension.

While symbolic links are recommended because they provide easy file system management and location transparency of database files, the drawback of using symbolic links is that you must manage two sets of files (for instance, during database backup and restore).

Usage notes

- If possible, use relative path names instead of absolute path names when creating symbolic links to access regular files as Quick I/O files. Using relative path names prevents copies of the symbolic link from referring to the original file when the directory is copied. This is important if you are backing up or moving database files with a command that preserves the symbolic link. However, some applications require absolute path names. If a file is then relocated to another directory, you must change the symbolic link to use the new absolute path. Alternatively, you can put all the symbolic links in a directory separate from the data directories. For example, you can create a directory named `/database` and put all the symbolic links there, with the symbolic links pointing to absolute path names.

To access an existing regular file as a Quick I/O file on a VxFS file system

- 1 Access the VxFS file system mount point containing the regular files:

```
$ cd /mount_point
```

- 2 Create the symbolic link:

```
$ mv filename .filename  
$ ln -s .filename::cdev:vxfs: filename
```

This example shows how to access the VxFS file `dbfile` as a Quick I/O file:

```
$ cd /db01  
$ mv dbfile .dbfile  
$ ln -s .dbfile::cdev:vxfs: dbfile
```

This example shows how to confirm the symbolic link was created:

```
$ ls -lo .dbfile dbfile  
  
-rw-r--r--  1 db2inst1 104890368 Oct 2 13:42  .dbfile  
  
lrwxrwxrwx  1 db2inst1    19      Oct 2 13:42  dbfile ->  
  .dbfile::vxcdev:vxfs:
```

Converting DB2 containers to Quick I/O files

Special commands available in the `/opt/VRTSdb2ed/bin` directory are provided to assist you in converting an existing database to use Quick I/O. You can use the `qio_getdbfiles` command to extract a list of file names from the database system

tables and the `qio_convertdbfiles` command to convert this list of database files to use Quick I/O.

Note: It is recommended that you create a Storage Checkpoint before converting to or from Quick I/O.

See [“Creating a Storage Checkpoint”](#) on page 275.

Before converting database files to Quick I/O files, the following conditions must be met:

- Prerequisites
- Log in as the DB2 instance owner (typically, the user ID `db2inst1`) to run the `qio_getdbfiles` and `qio_convertdbfiles` commands.
 - You must predefine the DB2 environment variable `$DB2DATABASE`.
 - The SFDB repository must exist before you can convert to Quick I/O files. Run the `db2ed_update` command to update or create the repository.
 - Files you want to convert must be regular files on VxFS file systems or links that point to regular VxFS files

Usage notes

- Converting existing database files to Quick I/O files may not be the best choice if the files are fragmented. Use the `-f` option to determine the fragmentation levels and choose one of two approaches: Either exclude files that are highly fragmented and do not have sufficient contiguous extents for Quick I/O use, or create new files with the `qiomkfile` command, rather than convert them with the `qio_convertdbfiles` command.
- If you choose to create new files, they will be contiguous. You must then move data from the old files to the new files using the DB2 database export/import facility `db2move` or `restore redirect`, and then define the new files to the database.
- `qio_getdbfiles` skips any tablespaces that have a type of system managed space (SMS), as these tablespaces are based on a directory format and not suitable for conversion.
The `qio_getdbfiles` command retrieves a list of database files and saves them in a file named `mkqio.dat` in the current directory.
- Instead of using the `qio_getdbfiles` command, you can manually create the `mkqio.dat` file containing the DB2 instance filenames that you want to convert to Quick I/O files.
- The `qio_convertdbfiles` command exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command.

The following options are available for the `qio_getdbfiles` command:

- T Lets you specify the type of database as `db2`. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `SYBASE`, `DSQUERY`, and `$DB2INSTANCE`, are present on a server).

The following options are available for the `qio_convertdbfiles` command:

- a Changes regular files to Quick I/O files using absolute path names. Use this option when symbolic links need to point to absolute path names (for example, at a site that uses SAP).
- h Displays a help message.

- T Lets you specify the type of database as `db2`. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `SYBASE`, `DSQUERY`, and `$DB2INSTANCE` are present on a server).
- u Changes Quick I/O files back to regular files. Use this option to undo changes made by a previous run of the `qio_convertdbfiles` script.

To extract a list of DB2 containers to convert

- ◆ With the database instance up and running, run the `qio_getdbfiles` command from a directory for which you have write permission:

```
$ cd /extract_directory
$ export DB2DATABASE=database_name
$ /opt/VRTSdb2ed/bin/qio_getdbfiles
```

The `qio_getdbfiles` command extracts the list file names from the database system tables and stores the file names and their size in bytes in a file called `mkqio.dat` under the current directory.

Note: Alternatively, you can manually create the `mkqio.dat` file containing the DB2 database container names that you want to convert to use Quick I/O. You can also manually edit the `mkqio.dat` file generated by `qio_getdbfiles`, and remove files that you do not want to convert to Quick I/O files.

Note: To run the `qio_getdbfiles` command, you must have permission to access the database and permission to write to the `/extract_directory`.

The `mkqio.dat` list file should look similar to the following:

```
/data11r1/VRTS11r1/redo01.log 52428800
/data11r1/VRTS11r1/redo02.log 52428800
/data11r1/VRTS11r1/redo03.log 52428800
/data11r1/VRTS11r1/sysaux01.dbf 632553472
/data11r1/VRTS11r1/system01.dbf 754974720
/data11r1/VRTS11r1/undotbs01.dbf 47185920
/data11r1/VRTS11r1/users01.dbf 5242880
/data11r1/nqio1.dbf 104857600
```

To convert the DB2 database files to Quick I/O files

- 1 Make the database inactive by either shutting down the instance or disabling user connections.

Warning: Running the `qio_convertdbfiles` command while the database is up and running can cause severe problems with your database, including loss of data and corruption.

- 2 Run the `qio_convertdbfiles` command from the directory containing the `mkqio.dat` file:

```
$ cd /extract_directory

$ export DB2DATABASE=database_name

$ /opt/VRTSdb2ed/bin/qio_convertdbfiles
```

The list of files in the `mkqio.dat` file is displayed. For example:

```
file1 --> .file1::cdev:vxfs:
file2 --> .file2::cdev:vxfs:
file3 --> .file3::cdev:vxfs:
file4 --> .file4::cdev:vxfs:
file5 --> .file5::cdev:vxfs:
```

Run the `qio_convertdbfiles` command (with no options specified) to rename the file *filename* to `.filename` and creates a symbolic link to `.filename` with the Quick I/O extension. By default, the symbolic link uses a relative path name.

The `qio_convertdbfiles` script exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command again.

- 3 Make the database active again.

You can now access these database files using the Quick I/O interface.

To undo the previous run of `qio_convertdbfiles` and change Quick I/O files back to regular VxFS files

- 1 If the database is active, make it inactive by either shutting down the instance or disabling user connections.
- 2 Run the following command from the directory containing the `mkqio.dat` file:

```
$ cd /extract_directory
$ export DB2DATABASE=database_name
$ /opt/VRTSdb2ed/bin/qio_convertdbfiles -u
```

The list of Quick I/O files in the `mkqio.dat` file is displayed. For example:

```
.file1::cdev:vxfs: --> file1
.file2::cdev:vxfs: --> file2
.file3::cdev:vxfs: --> file3
.file4::cdev:vxfs: --> file4
.file5::cdev:vxfs: --> file5
```

The `qio_convertdbfiles` command with the undo option (`-u`) specified renames the files from `<filename>` to `<filename>` and undoes the symbolic link to `.filename` that was created along with the Quick I/O files.

About sparse files

Support for sparse files lets applications store information (in inodes) to identify data blocks that have only zeroes, so that only blocks containing non-zero data have to be allocated on disk.

For example, if a file is 10KB, it typically means that there are blocks on disk covering the whole 10KB. Assume that you always want the first 9K to be zeroes. The application can go to an offset of 9KB and write 1KB worth of data. Only a block for the 1KB that was written is allocated, but the size of the file is still 10KB.

The file is now sparse. It has a hole from offset 0 to 9KB. If the application reads any part of the file within this range, it will see a string of zeroes.

If the application subsequently writes a 1KB block to the file from an offset of 4KB, for example, the file system will allocate another block.

The file then looks like:

- 0-4KB - hole
- 4-5KB - data block

- 5-9KB - hole
- 9-10KB - data block

So a 1TB file system can potentially store up to 2TB worth of files if there are sufficient blocks containing zeroes. Quick I/O files cannot be sparse and will always have all blocks specified allocated to them.

Displaying Quick I/O status and file attributes

You can obtain and display information about Quick I/O status and file attributes using various options of the `ls` command:

- al** Lists all files on a file system, including Quick I/O files and their links.
- lL** Shows if Quick I/O was successfully installed and enabled.
- a1L** Shows how a Quick I/O file name is resolved to that of a raw device.

To list all files on the current file system, including Quick I/O files and their links

- ◆ Use the `ls -al` command with the file names:

```
$ ls -al filename .filename
```

The following example shows how to use the `-a` option to display the absolute path name created using `qiomkfile`:

```
$ ls -al d* .d*

-rw-r--r--  1 db2inst1 db2iadm1 104890368  Oct 2 13:42  .dbfile

lrwxrwxrwx  1 db2inst1 db2iadm1   19      Oct 2 13:42  dbfile ->
.dbfile::cdev:vxfs:
```

To determine if Quick I/O is installed and enabled for DB2

- ◆ Use the `ls` command as follows:

```
$ ls -lL filename
```

The following example shows how to determine if Quick I/O is installed and enabled:

```
$ ls -lL dbfile
```

where the first character, `c`, indicates it is a raw character device file, and the major and minor device numbers are displayed in the size field. If you see a `No such file or directory` message, Quick I/O did not install properly or does not have a valid license key.

To show a Quick I/O file resolved to a raw device

- ◆ Use the `ls` command with the file names as follows:

```
$ ls -alL filename .filename
```

The following example shows how the Quick I/O file name `dbfile` is resolved to that of a raw device:

```
$ ls -alL d* .d*
```

```
crw-r--r--  1 db2inst1 db2iadml 45,  1          Oct 2 13:42  dbfile
-rw-r--r--  1 db2inst1 db2iadml 104890368    Oct 2 13:42  .dbfile
```

Extending a Quick I/O file in a DB2 environment

Although Quick I/O files must be preallocated, they are not limited to the preallocated sizes. You can grow or “extend” a Quick I/O file by a specific amount or to a specific size, using options to the `qiomkfile` command. Extending Quick I/O files is a fast, online operation and offers a significant advantage over using raw devices.

Before extending a Quick I/O file, make sure the following conditions have been met:

- Prerequisites
- You must have sufficient space on the file system to extend the Quick I/O file.

- Usage notes
- You can also grow VxFS file systems online (provided the underlying disk or volume can be extended) using the `fsadm` command. You can expand the underlying volume and the filesystem with the `vxresize` command.
 - You must have superuser (`root`) privileges to resize VxFS file systems using the `fsadm` command.
 - See the `fsadm_vxfs(1M)` and `qiomkfile(1M)` manual pages for more information.

The following options are available with the `qiomkfile` command:

- e Extends the file by a specified amount to allow DB2 container resizing.
- r Increases the file to a specified size to allow DB2 container resizing.

To extend a Quick I/O file

- 1 If required, ensure the underlying storage device is large enough to contain a larger VxFS file system (see the `vxassist(1M)` manual page for more information), and resize the VxFS file system using `fsadm` command:
- 2 Extend the Quick I/O file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -e extend_amount /mount_point/filename
```

or

```
$ /opt/VRTS/bin/qiomkfile -r newsize /mount_point/filename
```

An example to show how to grow VxFS file system:

/db01 to 500MB and extend the `tbs1_cont001` Quick I/O file by 20MB:

```
$ /opt/VRTS/bin/qiomkfile -e 20M /db01/tbs1_cont001
```

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

An example to show how to grow VxFS file system:

/db01 to 500MB and resize the `tbs1_cont001` Quick I/O file to 300MB:

```
$ /opt/VRTS/bin/qiomkfile -r 300M /db01/tbs1_cont001
```

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

Monitoring tablespace free space with DB2 and extending tablespace containers

DB2 does not automatically make use of extended DMS files. When tablespace space needs to be extended, a number of DB2 commands must be run. Unlike raw devices, a Database Administrator can easily extend Quick I/O files online. Using this method, a Database Administrator can monitor the free space available in the DB2 tablespaces and use the `qiomkfile` command to grow the Quick I/O files online as needed (typically when the file is about 80 to 90% full). This method does not require you to lock out unused disk space for Quick I/O files. The free space on the file system is available for use by other applications.

Before extending tablespaces, make sure the following conditions have been met:

- | | |
|---------------|--|
| Prerequisites | ■ Log on as the DB2 instance owner. |
| Usage notes | ■ Monitor the free space available in the Quick I/O file, and grow the file as necessary with the <code>qiomkfile</code> command.
■ A Database Administrator can grow underlying VxFS file systems online (provided the underlying disk or volume can be extended) using the <code>fsadm</code> command. See the <code>fsadm (1M)</code> manual page for more information.
■ It is strongly recommended that if a DB2 tablespace is running out of space, that all containers are grown by the same amount. It is possible to add extra containers to the tablespace, but this results in DB2 performing a relayout of the data to balance usage across all available containers in the tablespace.
Also refer to the DB2 Administration Guide section on <i>Managing Storage and the DB2 SQL Reference Guide</i> for information on the <code>ALTER TABLESPACE</code> command. |

To monitor the free space available in a DB2 tablespace

- ◆ Use the following DB2 commands:

```
$ db2 connect to database
$ db2 list tablespaces show detail
$ db2 terminate
```

To extend a Quick I/O file using `qiomkfile`

- ◆ Use the `qiomkfile` command to extend the Quick I/O file (if the container is running low on free blocks):

```
$ /opt/VRTS/bin/qiomkfile -e extend_amount filename
```

To extend a DB2 tablespace by a fixed amount

- ◆ Use the following DB2 commands:

```
$ db2 connect to database

$ db2 alter tablespace tablespace-name extend (ALL amount)

$ db2 terminate
```

This example shows how to monitor the free space on the tablespaces in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

This example shows how to extend the three DB2 containers owned by tablespace `EMP` by 500MB using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont001

$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont002

$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont003
```

This example shows how to notify DB2 that all containers in tablespace `EMP` have grown by 500MB:

```
$ db2 connect to PROD

$ db2 alter tablespace EMP extend (ALL 500M)

$ db2 terminate
```

This example shows how to verify the newly allocated space on the tablespace `EMP` in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

Recreating Quick I/O files after restoring a database

If you need to restore your database and were using Quick I/O files, you can use the `qio_recreate` command to automatically recreate the Quick I/O files after you have performed a full database recovery. The `qio_recreate` command uses the `mkqio.dat` file, which contains a list of the Quick I/O files used by the database and the file sizes.

For information on recovering your database, refer to the documentation that came with your database software.

Before recreating Quick I/O with the `qio_recreate` command, make sure the following conditions have been met:

- DB2 Prerequisites**
- Recover your database before attempting to recreate the Quick I/O files.
 - Log in as the DB2 instance owner (typically, the user ID `db2inst1`) to run the `qio_recreate` command.
 - In the directory from which you run the `qio_recreate` command, you must have an existing `mkqio.dat` file.
 - The `DB2DATABASE` environment variable must be set. See “[Converting DB2 containers to Quick I/O files](#)” on page 44.
- Usage notes**
- The `qio_recreate` command supports only conventional Quick I/O files.
 - Refer to the `qio_recreate(1M)` manual page for more information.

To recreate Quick I/O files after recovering a database

- ◆ As DBA, use the `qio_recreate` command as follows:

```
$ /opt/VRTSdb2ed/bin/qio_recreate
```

You will not see any output if the command is successful.

When you run the `qio_recreate` command, the following actions occur:

If...	Then...
a Quick I/O file is missing	the Quick I/O file is recreated.
a symbolic link from a regular VxFS file to a Quick I/O file is missing	the symbolic link is recreated.
a symbolic link and its associated Quick I/O file are missing	both the link and the Quick I/O file are recreated.

If...	Then...
a Quick I/O file is missing and the regular VxFS file that it is symbolically linked to is not the original VxFS file	the Quick I/O file is not recreated and a warning message is displayed.
a Quick I/O file is smaller than the size listed in the <code>mkqio.dat</code> file	the Quick I/O file is not recreated and a warning message is displayed.

Disabling Quick I/O

If you need to disable the Quick I/O feature, you need to remount the VxFS file system using a special mount option.

Warning: Do not disable Quick I/O on VxFS file systems containing Quick I/O containers of type . Doing so causes the tablespace containing them to go offline.

Before disabling Quick I/O, make sure the following condition has been met:

Prerequisite The file system you are planning to remount must be located in the `/etc/filesystems` file.

To remount the file system with Quick I/O disabled

- ◆ Use the command as follows:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio special  
/mount_point
```

The following example shows how to remount file system with Quick I/O disabled:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio \  
/dev/vx/dsk/dbdg/vol01  
/db01
```

Improving Sybase performance with Veritas Quick I/O

This chapter includes the following topics:

- [How to set up Quick I/O](#)
- [Preallocating space for Quick I/O files using the setext command](#)
- [Creating database files as Quick I/O files with qiomkfile](#)
- [Accessing regular VxFS files as Quick I/O files](#)
- [Converting Sybase files to Quick I/O files](#)
- [Displaying Quick I/O status and file attributes](#)
- [Extending a Quick I/O file in a Sybase environment](#)
- [Recreating Quick I/O files after restoring a database](#)
- [Disabling Quick I/O](#)

How to set up Quick I/O

Quick I/O is included in Veritas Storage Foundation Standard and Enterprise products. By default, Quick I/O is enabled when you mount a VxFS file system.

If Quick I/O is not available in the kernel, or a Veritas Storage Foundation Standard or Enterprise product license is not installed, a file system mounts without Quick I/O by default, the Quick I/O file name is treated as a regular file, and no error

message displays. If, however, you specify the `-o qio` option, the `mount` command prints an error and terminates without mounting the file system.

```
VxFDD: You don't have a license to run this program
vxfs mount: Quick I/O not available
```

Depending on whether you create a new database or convert an existing database to use Quick I/O, you have the following options:

To create a new Sybase database:

- Use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
See “[Creating database files as Quick I/O files with `qiomkfile`](#)” on page 59.
- Use the `setext` command to preallocate space for database files and create the Quick I/O files.
See “[Preallocating space for Quick I/O files using the `setext` command](#)” on page 58.

To convert an existing database:

- Create symbolic links for existing VxFS files, and use these symbolic links to access the files as Quick I/O files.
See “[Accessing regular VxFS files as Quick I/O files](#)” on page 62.
- Convert your existing Sybase database files to use the Quick I/O interface using the `qio_getdbfiles` and `qio_convertdbfiles` commands.
See “[Converting Sybase files to Quick I/O files](#)” on page 64.

Preallocating space for Quick I/O files using the `setext` command

As an alternative to using the `qiomkfile` command, you can also use the VxFS `setext` command to preallocate space for database files.

Before preallocating space with `setext`, verify the the following conditions are met:

- | | |
|---------------|---|
| Prerequisites | ■ The <code>setext</code> command requires superuser (<code>root</code>) privileges. |
| Usage notes | ■ You can use the <code>chown</code> command to change the owner and group permissions on the file after you create it.
See the <code>setext</code> (1M) manual page for more information. |

To create a Quick I/O database file using `setext`

- 1 Access the VxFS mount point and create a file:

```
# cd /mount_point
# touch .filename
```

- 2 Use the `setext` command to preallocate space for the file:

```
# /opt/VRTS/bin/setext -r size -f noreserve -f chgsize \
.filename
```

- 3 Create a symbolic link to allow databases or applications access to the file using its Quick I/O interface:

```
# ln -s .filename::cdev:vxfs: filename
```

- 4 Change the owner and group permissions on the file:

```
# chown sybase:sybase .filename
# chmod 660 .filename
```

An example to show how to access the mount point for Sybase `/db01`, create a datafile, preallocate the space, and change the permissions:

```
# cd /db01
# touch .dbfile
# /opt/VRTS/bin/setext -r 100M -f noreserve -f chgsize .dbfile
# ln -s .dbfile::cdev:vxfs: dbfile

# chown sybase:sybase .dbfile

# chmod 660 .dbfile
```

Creating database files as Quick I/O files with `qiomkfile`

The best way to preallocate space for tablespace containers and to make them accessible using the Quick I/O interface is to use the `qiomkfile`. You can use the `qiomkfile` to create the Quick I/O files for either temporary or permanent tablespaces.

Prerequisites	<ul style="list-style-type: none">■ You can create Quick I/O files only on VxFS file systems.■ To create device files on an existing file system, run <code>fsadm</code> (or similar utility) to report and eliminate fragmentation.■ You must have read/write permissions on the directory in which you intend to create DB2 Quick I/O files.
Usage notes	<ul style="list-style-type: none">■ The <code>qiomkfile</code> command creates two files: a regular file with preallocated, contiguous space, and a file that is a symbolic link pointing to the Quick I/O name extension.■ See the <code>qiomkfile(1M)</code> manual page for more information.
<code>-a</code>	Creates a symbolic link with an absolute path name for a specified file. Use the <code>-a</code> option when absolute path names are required. However, the default is to create a symbolic link with a relative path name.
<code>-e</code>	Extends a file by a specified amount to allow DB2 tablespace resizing. See “Extending a Quick I/O file in a Sybase environment” on page 72.
<code>-r</code>	Increases the file to a specified size to allow DB2 tablespace resizing. See “Extending a Quick I/O file in a Sybase environment” on page 72.
<code>-s</code>	Specifies the space to preallocate for a file in bytes, kilobytes, megabytes, gigabytes, or sectors (512 bytes) by adding a <code>k</code> , <code>K</code> , <code>m</code> , <code>M</code> , <code>g</code> , <code>G</code> , <code>s</code> , or <code>S</code> suffix. The default is bytes—you do not need to attach a suffix to specify the value in bytes. The size of the file that is preallocated is the total size of the file (including the header) rounded to the nearest multiple of the file system block size.

Warning: Exercise caution when using absolute path names. Extra steps may be required during database backup and restore procedures to preserve symbolic links. When you restore files to directories different from the original paths, you must change the symbolic links that use absolute path names to point to the new path names before you restart the database.

To create a database file as a Quick I/O file using qiomkfile

1 Create a database file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -s file_size /mount_point/filename
```

2 Add a device to the Sybase dataserver device pool for the Quick I/O file using the `disk init` command:

```
$ isql -Usa -Psa_password -Sdataserver_name
> disk init
> name="device_name",
> physname="/mount_point/filename",
> vdevno="device_number",
> size=51200
> go
> alter database production on new_device=file_size
> go
```

The size is in 2K units. The Enterprise Reference manual contains more information on the `disk init` command.

See the *Sybase Adaptive Server Enterprise Reference Manual*.

3 Use the file to create a new segment or add to an existing segment.

To add a new segment:

```
$ isql -Usa -Psa_password -Sdataserver_name
> sp_addsegment new_segment, db_name, device_name
> go
```

To extend a segment:

```
$ isql -Usa -Psa_password -Sdataserver_name
> sp_extendsegment segment_name, db_name, device_name
> go
```

See the *Sybase Adaptive Server Enterprise Reference Manual*.

An example to show how to create a 100MB database file named `dbfile` on the VxFS file system `/db01` using a relative path name:

```
$ /opt/VRTS/bin/qiomkfile -s 100m /db01/dbfile
$ ls -al
-rw-r--r--  1 sybase  sybase  104857600  Oct 2 13:42  .dbfile
lrwxrwxrwx  1 sybase  sybase      19      Oct 2 13:42  dbfile -> \
.dbfile::cdev:vxfs:
```

In the example, `qiomkfile` creates a regular file named `/db01/.dbfile`, which has the real space allocated. Then, `qiomkfile` creates a symbolic link named `/db01/dbfile`. The symbolic link is a relative link to the Quick I/O interface for `/db01/.dbfile`, that is, to the `.dbfile::cdev:vxfs:` file. The symbolic link allows `.dbfile` to be accessible to any database or application using its Quick I/O interface.

The device size is a multiple of 2K pages. In the example, 51200 times 2K pages is 104857600 bytes. The `qiomkfile` command must use this size.

An example to show how to add a 100MB Quick I/O file named `dbfile` to the list of devices used by database `production`, using the `disk init` command:

```
$ isql -Usa -Psa_password -Sdataserver_name
> disk init
> name="new_device",
> physname="/db01/dbfile",
> vdevno="device_number",
> size=51200
> go
> alter database production on new_device=100
> go
```

An example to show how to create a new segment, named `segment2`, for device `dbfile` on database `production`:

```
$ isql -Usa_password -Sdataserver_name
> sp_addsegment segment2, production, dbfile
> go
```

An example to show how to extend a segment, named `segment1`, for device `dbfile` on database `production`:

```
$ isql -Usa_password -Sdataserver_name
> sp_extendsegment segment1, production, dbfile
> go
```

Accessing regular VxFS files as Quick I/O files

You can access regular VxFS files as Quick I/O files using the `::cdev:vxfs:` name extension.

Symbolic links are recommended because they provide easy file system management and location transparency of database files. However, the drawback

of using symbolic links is that you must manage two sets of files (for instance, during database backup and restore).

Note: Sybase requires special prerequisites.

See [“Converting Sybase files to Quick I/O files”](#) on page 64.

- Usage notes
- If possible, use relative path names instead of absolute path names when creating symbolic links to access regular files as Quick I/O files. Using relative path names prevents copies of the symbolic link from referring to the original file when the directory is copied. This is important if you back up or move database files with a command that preserves the symbolic link. However, some applications require absolute path names. If a file is then relocated to another directory, you must change the symbolic link to use the new absolute path. Alternatively, you can put all the symbolic links in a directory separate from the data directories. For example, you can create a directory named `/database` and put all the symbolic links there, with the symbolic links pointing to absolute path names.

To access an existing regular file as a Quick I/O file on a VxFS file system

- 1 Access the VxFS file system mount point containing the regular files:

```
$ cd /mount_point
```

- 2 Create the symbolic link:

```
$ mv filename .filename
$ ln -s .filename::cdev:vxfs: filename
```

This example shows how to access the VxFS file `dbfile` as a Quick I/O file:

```
$ cd /db01
$ mv dbfile .dbfile
$ ln -s .dbfile::cdev:vxfs: dbfile
```

This example shows how to confirm the symbolic link was created:

```
$ ls -lo .dbfile dbfile

-rw-r--r--  1 sybase  104890368  Oct 2 13:42  .dbfile

lrwxrwxrwx  1 sybase      19      Oct 2 13:42  dbfile ->
.dbfile::cdev:vxfs:
```

Converting Sybase files to Quick I/O files

Special commands are provided to assist you in identifying and converting an existing database to use Quick I/O. Use the `qio_getdbfiles` and `qio_convertdbfiles` commands to first extract and then convert Sybase dataserver files to Quick I/O files.

- Prerequisites
- Log in as the Database Administrator (typically, the user ID `sybase`) to run the `qio_getdbfiles` and `qio_convertdbfiles` commands.
 - Files you want to convert must be regular VxFS files created by Sybase ASE 12.0 or later and have the Sybase `dsync` flag on. Regular VxFS files created without the `dsync` flag on may be sparse. Sparse files should not be converted to Quick I/O files.
 - Sybase ASE Server must be installed and running.
 - The conversion commands were developed to support localization, so you need to set the `NLS_PATH` to obtain the proper message catalog.
 - For Sybase ASE 12.0, the `$$SYBASE` environment variable must be set to the Sybase home directory
For Sybase ASE 12.0, the `$$DSQUERY` environment variable must be set to the server on which you intend to run the `qio_getdbfiles` command
For Sybase ASE 12.0, the `$$PATH` environment variable must include `$$SYBASE/ASE-12_0/bin` and `$$SYBASE/OCS-12_0/bin`
For Sybase ASE 12.0, the `$$LD_LIBRARY_PATH` environment variable must include `$$SYBASE/OCS-12_0/lib`
For Sybase ASE 12.5, you must set the following Sybase environment variables:
`$$SYBASE` must be set to the Sybase home directory
`$$DSQUERY` must be set to the server on which you intend to run the `qio_getdbfiles` command
`$$PATH` must include `$$SYBASE/ASE-12_5/bin` and `$$SYBASE/OCS-12_5/bin`
`$$LD_LIBRARY_PATH` must include `$$SYBASE/OCS-12_5/lib`
Only English is supported in the current release.

For the `qio_getdbfiles` command:

- `-d` Enables you to specify a specific database name from which to extract a list of dataserver files. If you do not specify the `-d` option, the list of files comes from all databases of the Sybase server. You cannot use this option with the `-m` option.

- `-m` Enables you to specify a master device path. A master device does not have a corresponding physical path name in Sybase's database catalog, but rather has a `d_master` string. When you start an ASE server, you must pass in the full path name of master device. The `qio_getdbfiles` command first attempts to get the master device path for the Sybase in the `RUN_server_name` file in the `$$SYBASE/ASE-12_0/install` directory for Sybase ASE 12.0 or `$$SYBASE/ASE-12_5/install` directory for Sybase ASE 12.5 automatically. However, if you do not have a standard `RUN_server_name` file, you can use `-m` flag to pass in the master device path. You cannot use this option with the `-d` option.
- `-T` Enables you to specify the type of database as `syb`. Use this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `$$SYBASE`, and `$$DSQUERY`, are present on a server).

For the `qio_convertdbfiles` command:

- `-a` Changes regular files to Quick I/O files using absolute path names. Use this option when symbolic links need to point to absolute path names.
- `-f` Reports on the current fragmentation levels for database files listed in the `mkqio.dat` file. Fragmentation levels display as not fragmented, slightly fragmented, fragmented, highly fragmented. You must be superuser (`root`) to use this option.
- `-h` Displays a help message.
- `-T` Lets you specify the type of database as `syb`. Use this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `$$SYBASE`, and `$$DSQUERY`, are present on a server).
- `-u` Changes Quick I/O files back to regular files. Use this option to reverse changes made by a previous run of the `qio_convertdbfiles` script.

For converting Sybase files to Quick I/O files:

- The `qio_getdbfiles` and `qio_convertdbfiles` commands access the Sybase ASE Server to obtain information. You need to do one of the following to connect to the Sybase ASE Server:
- Supply the Sybase `sa` password when prompted (default behavior).

- Create a file named `sa_password_dataserver_name` (where `dataserver_name` is server defined in the `$DSQUERY` environment variable) in the `/opt/VRTSsybed/.private` directory which contains the sa password. The `.private` directory must be owned by the Sybase database administrator user (typically `sybase`) and carry a file permission mode of `700`. The `sa_password_dataserver_name` file must also be owned by the Sybase database administrator user, and carry a file permission mode of `600`. Once the `sa_password_dataserver_name` file is correctly set up, users will not be prompted for the sa password, and the convert commands will run in non-interactive mode.

For the `qio_getdbfiles` command:

- You can use the `qio_getdbfiles` command to generate lists of files from one or more databases.

For the `qio_convertdbfiles` command:

- To use the `qio_convertdbfiles` command with the `-f` option (the option that lets you report fragmentation), you must be superuser (`root`).
- Converting existing database files to Quick I/O files may not be the optimal thing to do if these files are fragmented. Use the `-f` option to determine the fragmentation levels and either:
 - Exclude files which are highly fragmented and do not have sufficient contiguous extents for Quick I/O use.
 - Create new files with the `qiomkfile` command, rather than converting the files using the `qio_convertdbfiles` command. The new files are contiguous. When the new files are created, you can move data from the old files to the new files using the `dd(1M)` command or a database import facility and the new files defined to the database.

To set the Sybase environment variables and NLSPATH

- ◆ Set the required Sybase environment and message catalog variables, as follows:

```
$ SYBASE=/home_directory; export SYBASE
$ DSQUERY=servername; export DSQUERY
$ PATH=$SYBASE/ASE-12_5/bin:$SYBASE/OCS-12_5/bin:$PATH; \
  export PATH
$ LD_LIBRARY_PATH=$SYBASE/OCS-12_5/lib; export LD_LIBRARY_PATH
$ NLSPATH=/usr/lib/locale/%L/%N:$NLSPATH; export NLSPATH
```

To determine if the Sybase server is up and running

- 1 Access the `install` directory:

```
$ cd $SYBASE/ASE-12_5/install
```

- 2 Use the `showserver` and `grep` commands to determine if the Sybase server is running:

```
$ ./showserver | grep servername
```

If the output of these commands displays the server name, the server is running. If the no output is displayed, the server is not running.

To extract a list of Sybase files to convert

- 1 Supply the sa password when prompted, or create a file named `sa_password_<dataserver_name>` in the `/opt/VRTSsybed/.private` directory which contains the sa password.
- 2 When the Sybase dataserver is up and running, run the `qio_getdbfiles` command without the `-d` option (to extract a list of dataserver files on all databases) from a directory for which you have write permission:

```
$ cd /extract_directory  
$ /opt/VRTSsybed/bin/qio_getdbfiles
```

or

When the database instance is up and running, run the `qio_getdbfiles` command with the `-d` option (to extract a list of dataserver files on a specific database) from a directory for which you have write permission:

```
$ cd /extract_directory  
$ /opt/VRTSsybed/bin/qio_getdbfiles -d database_name
```

The `qio_getdbfiles` command extracts the list of dataserver files and stores the file names in a file called `mkqio.dat`.

Note: Alternatively, you can manually create the `mkqio.dat` file containing the Sybase dataserver file names that you want to convert to use Quick I/O. You can also manually edit the `mkqio.dat` list file generated, and remove any files that you do not want to convert to Quick I/O files.

To convert the Sybase files to Quick I/O files

- 1 Shut down the Sybase dataserver.

Caution: Running the `qio_convertdbfiles` command while the database is up and running can cause severe problems with your database, including loss of data, and corruption.

- 2 Supply the sa password when prompted, or create a file named `sa_password_dataserver_name` in the `/opt/VRTSsybed/.private` directory which contains the sa password.
- 3 Run the `qio_convertdbfiles` command from the directory containing the `mkqio.dat` file:

```
$ cd /extract_directory
$ /opt/VRTSsybed/bin/qio_convertdbfiles
```

The list of files in the `mkqio.dat` file is displayed, for example:

```
    /sybdev/L001/SYS/master.dat
104857600
           /sybdev/L001/USER/qiofile1
209715200
           /sybdev/L001/USER/qiofile2
209715200
           /sybdev/L001/USER/qiofile3
209715200
           /sybdev/L001/USER/qiofile4
209715200
           /sybdev/L001/SYS/sysporcs.dat
83886080
```

The `qio_convertdbfiles` command (with no options specified) renames the file `<filename>` to `.<filename>` and creates a symbolic link to `.<filename>` with the Quick I/O extension. By default, the symbolic link uses a relative path name.

The `qio_convertdbfiles` command and prints an error message if any of the dataserver files are not on a VxFS file system. If this error occurs, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command again.

- 4 Start the database.

You can now access these dataserver files using the Quick I/O interface.

- To convert the database files listed in the `mkqio.dat` file to Quick I/O files, shut down the database and enter:

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles
```

```
Check whether Sybase server L001 is up running...
```

```
Attempt to Connect to Server L001...
```

```
Enter Sybase SA password for Server L001:
```

```
Password: pel93
```

```
CT-LIBRARY error:
```

```
ct_connect(): network packet layer: internal net library error:  
Net-Lib protocol driver call to connect two endpoints failed
```

Note: This error message is displayed because the `dataserver` is shut down. This behavior is the correct, expected behavior.

```
master.dat --> .master.dat::cdev:vxfs:
```

```
qiofile1 --> .qiofile1::cdev:vxfs:
```

```
qiofile2 --> .qiofile2::cdev:vxfs:
```

```
qiofile3 --> .qiofile3::cdev:vxfs:
```

```
qiofile4 --> .qiofile4::cdev:vxfs:
```

```
sysporcs.dat --> .sysporcs.dat::cdev:vxfs:
```

- To undo the previous run of `qio_convertdbfiles`, changing Quick I/O files back to regular VxFS files:

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles -u
```

```
.master.dat::cdev:vxfs: --> master.dat
```

```
.qiofile1::cdev:vxfs: --> qiofile1
```

```
.qiofile2::cdev:vxfs: --> qiofile2
```

```
.qiofile3::cdev:vxfs: --> qiofile3
```

```
.qiofile4::cdev:vxfs: --> qiofile4
```

```
.sysporcs.dat::cdev:vxfs: --> sysporcs.dat
```

Note: If the server is up and running, you receive an error message stating that you need to shut down before you can run the `qio_convertdbfiles` command.

Displaying Quick I/O status and file attributes

You can obtain and display information about Quick I/O status and file attributes using various options of the `ls` command:

- al** Lists all files on a file system, including Quick I/O files and their links.
- lL** Shows if Quick I/O was successfully installed and enabled.
- a1L** Shows how a Quick I/O file name is resolved to that of a raw device.

To list all files on the current file system, including Quick I/O files and their links

- ◆ Use the `ls -al` command with the file names:

```
$ ls -al filename .filename
```

The following example shows how to use the `-a` option to display the absolute path name which was created using `qiomkfile`:

```
$ ls -al d* .d*
-rw-r--r--  1 sybase  sybase  104890368  Oct 2 13:42  .dbfile
lrwxrwxrwx  1 sybase  sybase    19      Oct 2 13:42  dbfile ->
.dbfile::cdev:vxfs:
```

To determine if a Sybase segment has been converted to Quick I/O

- ◆ Use the `ls` command as follows:

```
$ ls -lL filename
```

The following example shows how to determine if Quick I/O is installed and enabled:

```
$ ls -lL dbfile
crw-r--r--  1 sybase  dba  45, 1  Oct 2 13:42  dbfile
```

To show a Quick I/O file resolved to a raw device

- ◆ Use the `ls` command with the file names as follows:

```
$ ls -all filename .filename
```

The following example shows how the Quick I/O file name `dbfile` is resolved to that of a raw device:

```
$ ls -all d* .d*
```

```
crw-r--r--  1 sybase  sybase   45,  1          Oct 2 13:42  dbfile
-rw-r--r--  1 sybase  sybase 104890368      Oct 2 13:42  .dbfile
```

Extending a Quick I/O file in a Sybase environment

Although Quick I/O files must be preallocated, they are not limited to the preallocated sizes. You can grow or “extend” a Quick I/O file by a specific amount or to a specific size, using options to the `qiomkfile` command. Extending Quick I/O files is a fast, online operation and offers a significant advantage over using raw devices.

Before extending a Quick I/O file, verify the following conditions are met:

- | | |
|---------------|--|
| Prerequisites | ■ You must have sufficient space on the file system to extend the Quick I/O file. |
| Usage notes | ■ You can also grow VxFS file systems online (provided the underlying disk or volume can be extended) using the <code>fsadm</code> command. You can expand the underlying volume and the file system with the <code>vxresize</code> command.
■ You must have superuser (<code>root</code>) privileges to resize VxFS file systems using the <code>fsadm</code> command.
■ For Sybase: you have the ability to extend a Quick I/O file, you cannot resize a database device in Sybase once it is initialized. However, with the ability to grow the volumes and file systems online, you can easily allocate new database devices to be used for new segments and to extend existing segments.
■ See the <code>fsadm_vxfs</code> (1M) and <code>qiomkfile</code> (1M) manual pages for more information. |

The following options are available with the `qiomkfile` command:

- | | |
|-----------------|--|
| <code>-e</code> | Extends the file by a specified amount to allow Sybase resizing. |
|-----------------|--|

`-r` Increases the file to a specified size to allow Sybase resizing.

To extend a Quick I/O file

- 1 If required, verify the underlying storage device is large enough to contain a larger VxFS file system (see the `vxassist(1M)` manual page for more information), and resize the VxFS file system using `fsadm` command.

For Sybase, for example:

```
# /opt/VRTS/bin/fsadm -b newsize /mount_point
```

where:

- `-b` is the option for changing the file size
 - `newsize` is the new size of the file system in bytes, kilobytes, megabytes, blocks, or sectors
 - `mount_point` is the file system's mount point
- 2 Extend the Quick I/O file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -e extend_amount /mount_point/filename
```

or

```
$ /opt/VRTS/bin/qiomkfile -r newsize /mount_point/filename
```

An example to show how to grow VxFS file system:

An example to show how to grow VxFS file system `/db01` to 500MB and extend the `dbfile` Quick I/O file by 20MB:

```
$ /opt/VRTS/bin/qiomkfile -e 20M /db01/dbfile
```

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

An example to show how to grow VxFS file system:

`/db01` to 500MB and resize the `dbfile` Quick I/O file to 300MB:

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

```
$ /opt/VRTS/bin/qiomkfile -r 300M /db01/dbfile
```

Recreating Quick I/O files after restoring a database

If you need to restore your database and you were using Quick I/O files, you can use the `qio_recreate` command to automatically recreate the Quick I/O files after you have performed a full database recovery. The `qio_recreate` command uses the `mkqio.dat` file, which contains a list of the Quick I/O files used by the database and the file sizes.

For information on recovering your database, refer to the documentation that came with your database software.

Before recreating Quick I/O with the `qio_recreate` command, verify the following conditions are met:

- | | |
|----------------------|---|
| Sybase Prerequisites | <ul style="list-style-type: none">■ Recover your database before attempting to recreate the Quick I/O files.■ Log in as the Database Administrator (typically, the user ID <code>sybase</code>) to run the <code>qio_recreate</code> command.■ In the directory from which you run the <code>qio_recreate</code> command, you must have an existing <code>mkqio.dat</code> file.■ The <code>SYBASE</code> and <code>DSQUERY</code> environment variables must be set. See “Converting Sybase files to Quick I/O files” on page 64. |
| Usage notes | <ul style="list-style-type: none">■ The <code>qio_recreate</code> command supports only conventional Quick I/O files.■ Refer to the <code>qio_recreate(1M)</code> manual page for more information. |

To recreate Quick I/O files after recovering a database

- ◆ As DBA, use the `qio_recreate` command as follows:

```
$ /opt/VRTSsybed/bin/qio_recreate
```

You do not see any output if the command is successful.

When you run the `qio_recreate` command, the following actions occur:

If...	Then...
a Quick I/O file is missing	the Quick I/O file is recreated.
a symbolic link from a regular VxFS file to a Quick I/O file is missing	the symbolic link is recreated.
a symbolic link and its associated Quick I/O file are missing	both the link and the Quick I/O file are recreated.

If...	Then...
a Quick I/O file is missing and the regular VxFS file that it is symbolically linked to is not the original VxFS file	the Quick I/O file is not recreated and a warning message is displayed.
a Quick I/O file is smaller than the size listed in the <code>mkqio.dat</code> file	the Quick I/O file is not recreated and a warning message is displayed.

Disabling Quick I/O

Before disabling Quick I/O, verify the following condition is met:

Prerequisite The file system you plan to remount must be located in the `/etc/filesystems` file.

To disable Quick I/O

- 1 If the database is running, shut it down.
- 2 To change Quick I/O files back to regular VxFS files, run the following command from the directory containing the `mkqio.dat` list:

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles -u
```

The list of Quick I/O files in the `mkqio.dat` file is displayed. For example:

```
.file1::cdev:vxfs: --> file1
.file2::cdev:vxfs: --> file2
.file3::cdev:vxfs: --> file3
.file4::cdev:vxfs: --> file4
.file5::cdev:vxfs: --> file5
```

The `qio_convertdbfiles` command with the undo option (`-u`) renames the files from *filename* to *filename* and removes the symbolic link to *filename* that was created along with the Quick I/O files.

- 3 To remount the file system with Quick I/O disabled, use the `mount -o noqio` command as follows:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio /mount_point
```


Improving DB2 database performance with Veritas Cached Quick I/O

This chapter includes the following topics:

- [Tasks for setting up Cached Quick I/O](#)
- [Enabling Cached Quick I/O on a file system](#)
- [Determining candidates for Cached Quick I/O](#)
- [Enabling and disabling Cached Quick I/O for individual files](#)

Tasks for setting up Cached Quick I/O

To set up and use Cached Quick I/O, you should do the following in the order in which they are listed:

- Enable Cached Quick I/O on the underlying file systems used for your database.
- Exercise the system in your production environment to generate file I/O statistics.
- Collect the file I/O statistics while the files are in use.
- Analyze the file I/O statistics to determine which files benefit from Cached Quick I/O.
- Disable Cached Quick I/O on files that do not benefit from caching.

Enabling Cached Quick I/O on a file system

Cached Quick I/O depends on Veritas Quick I/O running as an underlying system enhancement in order to function correctly. Follow the procedures listed here to ensure that you have the correct setup to use Cached Quick I/O successfully.

Prerequisites

- You must have permission to change file system behavior using the `vxtunefs` command to enable or disable Cached Quick I/O. By default, you need superuser (`root`) permissions to run the `vxtunefs` command, but other system users do not. Superuser (`root`) must specifically grant database administrators permission to use this command as follows:

```
# chown root:db2iadm1 /opt/VRTS/bin/vxtunefs
```

```
# chmod 4550 /opt/VRTS/bin/vxtunefs
```

where users belonging to the `db2iadm1` group are granted permission to run the `vxtunefs` command. We recommend this selective, more secure approach for granting access to powerful commands.

- You must enable Quick I/O on the file system. Quick I/O is enabled automatically at file system mount time.

If you have correctly enabled Quick I/O on your system, you can proceed to enable Cached Quick I/O as follows:

- Set the file system Cached Quick I/O flag, which enables Cached Quick I/O for all files in the file system.
- Setting the file system Cached Quick I/O flag enables caching for all files in the file system. You must disable Cached Quick I/O on individual Quick I/O files that do not benefit from caching to avoid consuming memory unnecessarily. This final task occurs at the end of the enabling process.

Usage notes

- If Cached Quick I/O is enabled, it is recommended that you monitor any paging activity to the swap device on your database servers. You can use the `vmstat -I` command to monitor swap device paging. If swap device paging is observed, proper AIX Virtual Memory Manager (VMM) tuning is required to improve database performance.

Enabling and disabling the `qio_cache_enable` flag

As superuser (`root`), set the `qio_cache_enable` flag using the `vxtunefs` command after you mount the file system.

To enable the `qio_cache_enable` flag for a file system

- ◆ Use the `vxtunefs` command as follows:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /db02
```

where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to “1” enables Cached Quick I/O. This command enables caching for all the Quick I/O files on this file system.

To disable the flag on the same file system

- ◆ Use the `vxtunefs` command as follows:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /db02
```

where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to “0” disables Cached Quick I/O. This command disables caching for all the Quick I/O files on this file system.

Making Cached Quick I/O settings persistent across reboots and mounts

You can make the Cached Quick I/O system setting persistent across reboots and mounts by adding a file system entry in the `/etc/vx/tunefstab` file.

Note: The `tunefstab` file is a user-created file. For information on how to create the file and add tuning parameters, see the `tunefstab(4)` manual page.

To enable a file system after rebooting

- ◆ Put the file system in the `/etc/vx/tunefstab` file and set the flag entry:

```
/dev/vx/dsk/dgname/volname qio_cache_enable=1
```

where:

- `/dev/vx/dsk/dgname/volname` is the name of a block device
- `dgname` is the name of the disk group
- `volname` is the name of the volume

For example:

```
/dev/vx/dsk/PRODDg/db01 qio_cache_enable=1  
/dev/vx/dsk/PRODDg/db02 qio_cache_enable=1
```

where `/dev/vx/dsk/PRODDg/db01` is the block device on which the file system resides.

The `tunefstab` (4) manual pages contain information on how to add tuning parameters.

See the `tunefstab` (4) manual page.

Note: `vxtunefs` can specify a mount point or a block device; `tunefstab` must always specify a block device only.

Using `vxtunefs` to obtain tuning information

Check the setting of the `qio_cache_enable` flag for each file system using the `vxtunefs` command.

To obtain information on only the `qio_cache_enable` flag setting

- ◆ Use the `grep` command with `vxtunefs`:

```
# /opt/VRTS/bin/vxtunefs /mount_point | grep qio_cache_enable
```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01 | grep qio_cache_enable
```

where `/db01` is the name of the file system. This command displays only the `qio_cache_enable` setting as follows:

```
qio_cache_enable = 0
```

You can also use the `vxtunefs` command to obtain a more complete list of I/O characteristics and tuning statistics.

See the `vxtunefs` (1) manual page.

To obtain information on all `vxtunefs` system parameters

- ◆ Use the `vxtunefs` command without `grep`:

```
# /opt/VRTS/bin/vxtunefs /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01
```

The `vxtunefs` command displays output similar to the following:

```
Filesystem i/o parameters for /db01
read_pref_io = 2097152
read_nstream = 1
read_unit_io = 2097152
write_pref_io = 2097152
write_nstream = 1
write_unit_io = 2097152
pref_strength = 10
buf_breakup_size = 2097152
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 1
write_throttle = 0
max_diskq = 33554432
initial_extent_size = 8
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 222425088
fcl_keeptime = 0
fcl_winterval = 3600
fcl_ointerval = 600
oltp_load = 0
```

The `vxtunefs(1)` manual pages contain a complete description of `vxtunefs` parameters and the tuning instructions.

See the `vxtunefs(1)` manual page.

Determining candidates for Cached Quick I/O

Determining which files can benefit from Cached Quick I/O is an iterative process that varies with each application. For this reason, you may need to complete the following steps more than once to determine the best possible candidates for Cached Quick I/O.

Before determining candidate files for Quick I/O, make sure the following conditions have been met:

- Prerequisites ■ You must enable Cached Quick I/O for the file systems.
 See [“Enabling Cached Quick I/O on a file system”](#) on page 78.
- Usage notes ■ See the `qiostat` (1M) manual page for more information.

Collecting I/O statistics

Once you have enabled Cached Quick I/O on a file system, you need to collect statistics to determine and designate the files that can best take advantage of its benefits.

To collect statistics needed to determine files that benefit from Cached Quick I/O

- 1 Reset the `qiostat` counters by entering:

```
$ /opt/VRTS/bin/qiostat -r /mount_point/filenames
```

- 2 Run the database under full normal load and through a complete cycle (24 to 48 hours in most cases) to determine your system I/O patterns and database traffic in different usage categories (for example, OLTP, reports, and backups) at different times of the day.
- 3 While the database is running, run `qiostat -l` to report the caching statistics as follows:

```
$ /opt/VRTS/bin/qiostat -l /mount_point/filenames
```

or, use the `-i` option to see statistic reports at specified intervals:

```
$ /opt/VRTS/bin/qiostat -i n /mount_point/filenames
```

where `n` is time in seconds

For example:

To collect I/O statistics from all database containers on file system `/db01:`

```
$ /opt/VRTS/bin/qiostat -l /db01/*
```

About I/O statistics

The output of the `qiostat` command is the primary source of information to use in deciding whether to enable or disable Cached Quick I/O on specific files. Statistics are printed in two lines per object.

The second line of information is defined as follows:

- **CREAD** is the number of reads from the VxFS cache (or total number of reads to Quick I/O files with cache advisory on)
- **PREAD** is the number of reads going to the disk for Quick I/O files with the cache advisory on
- **HIT_RATIO** is displayed as a percentage and is the number of **CREADS** minus the number of **PREADS** times 100 divided by the total number of **CREADS**. The formula looks like this:

$$(\text{CREADS} - \text{PREADS}) * 100 / \text{CREADS}$$

The `qiostat -l` command output looks similar to the following:

```
/db01/tbs1_cont001 6          1      21      4      10.0  0.0
                   6          6      0.0

/db01/tbs2_cont001 62552 38498 250213 153992  21.9  0.4
62567 49060      21.6

/db01/tbs2_cont002 62552 38498 250213 153992  21.9  0.4
62567 49060      21.6
```

Analyze the output to find out where the cache-hit ratio is above a given threshold. A cache-hit ratio above 20 percent on a file for a given application may be sufficient to justify caching on that file. For systems with larger loads, the acceptable ratio may be 30 percent or above. Cache-hit-ratio thresholds vary according to the database type and load.

Using the sample output above as an example, the file `/db01/tbs1_cont001` does not benefit from the caching because the cache-hit ratio is zero. In addition, the file receives very little I/O during the sampling duration.

However, the files `/db01/tbs2_*` have a cache-hit ratio of 21.6 percent. If you have determined that, for your system and load, this figure is above the acceptable threshold, it means the database can benefit from caching. Also, study the numbers reported for the read and write operations. When you compare the number of reads and writes for the `/db01/tbs2_*` files, you see that the number of reads is roughly twice the number of writes. You can achieve the greatest performance gains with Cached Quick I/O when using it for files that have higher read than write activity.

Based on these two factors, `/db01/tbs2_cont001` and `/db01/tbs2_cont002` are prime candidates for Cached Quick I/O.

See [“Enabling and disabling Cached Quick I/O for individual files”](#) on page 84.

Effects of read-aheads on I/O statistics

The number of `CREADS` in the `qiostat` output is the total number of reads performed, including Cached Quick I/O, and the number of `PREADS` is the number of physical reads. The difference between `CREADS` and `PREADS` (`CREADS - PREADS`) is the number of reads satisfied from the data in the file system cache. Thus, you expect that the number of `PREADS` would always be equal to or lower than the number of `CREADS`.

However, the `PREADS` counter also increases when the file system performs read-aheads. These read-aheads occur when the file system detects sequential reads. In isolated cases where cache hits are extremely low, the output from `qiostat` could show that the number of `CREADS` is lower than the number of `PREADS`. The cache-hit ratio calculated against these `CREAD/PREAD` values is misleading when used to determine whether Cached Quick I/O should be enabled or disabled.

Under these circumstances, you can make a more accurate decision based on a collective set of statistics by gathering multiple sets of data points. Consequently, you might want to enable Cached Quick I/O for all the container files that contain a particular table, across multiple tablespaces used by a given database, even if the containers in just one of the tablespaces exhibited a high cache hit ratio. In general, we expect all containers in a tablespace to have approximately the same cache hit ratio.

Other tools for analysis

While the output of the `qiostat` command is the primary source of information to use in deciding whether to enable Cached Quick I/O on specific files, we also recommend using other tools in conjunction with `qiostat`. For example, benchmarking software that measures database throughput is also helpful. If a benchmark test in which Cached Quick I/O was enabled for a certain set of data files resulted in improved performance, you can also use those results as the basis for enabling Cached Quick I/O.

Enabling and disabling Cached Quick I/O for individual files

After using `qiostat` or other analysis tools to determine the appropriate files for Cached Quick I/O, you need to disable Cached Quick I/O for those individual files that do not benefit from caching using the `qioadmin` command.

- | | |
|---------------|---|
| Prerequisites | ■ Enable Cached Quick I/O for the file system before enabling or disabling Cached Quick I/O at the individual file level. |
| Usage notes | ■ You can enable or disable Cached Quick I/O for individual files while the database is online.
■ You should monitor files regularly using <code>qiostat</code> to ensure that a file's cache-hit ratio has not changed enough to reconsider enabling or disabling Cached Quick I/O for the file.
■ Enabling or disabling Cached Quick I/O for an individual file is also referred to as setting the cache advisory on or off.
■ See the <code>qioadmin(1)</code> manual page. |

Setting cache advisories for individual files

You can enable and disable Cached Quick I/O for individual files by changing the cache advisory settings for those files.

To disable Cached Quick I/O for an individual file

- ◆ Use the `qioadmin` command to set the cache advisory to `OFF` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=OFF /mount_point
```

For example, to disable Cached Quick I/O for the file `/db01/dbfile`, set the cache advisory to `OFF`:

```
$ /opt/VRTS/bin/qioadmin -S dbfile=OFF /db01
```

To enable Cached Quick I/O for an individual file

- ◆ Use the `qioadmin` command to set the cache advisory to `ON` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=ON /mount_point
```

For example, running `qiostat` shows the cache hit ratio for the file `/db01/dbfile` reaches a level that would benefit from caching. To enable Cached Quick I/O for the file `/db01/dbfile`, set the cache advisory to `ON`:

```
$ /opt/VRTS/bin/qioadmin -S dbfile=ON /db01
```

Making individual file settings for Cached Quick I/O persistent

You can make the enable or disable individual file settings for Cached Quick I/O persistent across reboots and mounts by adding cache advisory entries in the `/etc/vx/qioadmin` file.

Cache advisories set using the `qioadmin` command are stored as extended attributes of the file in the inode. These settings persist across file system remounts and system reboots, but these attributes are not backed up by the usual backup methods, so they cannot be restored. Therefore, always be sure to reset cache advisories after each file restore. This is not necessary if you maintain the cache advisories for Quick I/O files in the `/etc/vx/qioadmin` file.

To enable or disable individual file settings for Cached Quick I/O automatically after a reboot or mount

- ◆ Add cache advisory entries in the `/etc/vx/qioadmin` file as follows:

```
device=/dev/vx/dsk/<diskgroup>/<volume>

filename,OFF

filename,OFF

filename,OFF

filename,ON
```

For example, to make the Cached Quick I/O settings for individual files in the `/db01` file system persistent, edit the `/etc/vx/qioadmin` file similar to the following:

```
#
# List of files to cache in /db01 file system
#
device=/dev/vx/dsk/PRODDg/db01

dbfile01,OFF
dbfile02,OFF
dbfile03,ON
```

Determining individual file settings for Cached Quick I/O using `qioadmin`

You can determine whether Cached Quick I/O is enabled or disabled for individual files by displaying the file's cache advisory setting using the `qioadmin` command.

Note: To verify caching, always check the setting of the flag `qio_cache_enable` using `vxtunefs`, along with the individual cache advisories for each file.

To display the current cache advisory settings for a file

- ◆ Use the `qioadmin` command with the `-P` option as follows:

```
$ /opt/VRTS/bin/qioadmin -P filename /mount_point
```

For example, to display the current cache advisory setting for the file `dbfile` in the `/db01` file system:

```
$ /opt/VRTS/bin/qioadmin -P dbfile /db01
```

```
dbfile,OFF
```


Improving Sybase database performance with Veritas Cached Quick I/O

This chapter includes the following topics:

- [Tasks for setting up Cached Quick I/O](#)
- [Enabling Cached Quick I/O on a file system](#)
- [Determining candidates for Cached Quick I/O](#)
- [Enabling and disabling Cached Quick I/O for individual files](#)

Tasks for setting up Cached Quick I/O

To set up and use Cached Quick I/O, you should do the following in the order in which they are listed:

- Enable Cached Quick I/O on the underlying file systems used for your database.
- Exercise the system in your production environment to generate file I/O statistics.
- Collect the file I/O statistics while the files are in use.
- Analyze the file I/O statistics to determine which files benefit from Cached Quick I/O.
- Disable Cached Quick I/O on files that do not benefit from caching.

Enabling Cached Quick I/O on a file system

Cached Quick I/O depends on Veritas Quick I/O running as an underlying system enhancement in order to function correctly. Follow the procedures listed here to ensure that you have the correct setup to use Cached Quick I/O successfully.

- Usage notes
- If Cached Quick I/O is enabled, it is recommended that you monitor any paging activity to the swap device on your database servers. You can use the `vmstat -I` command to monitor swap device paging. If swap device paging is observed, proper AIX Virtual Memory Manager (VMM) tuning is required to improve database performance.

Enabling and disabling the `qio_cache_enable` flag

As superuser (`root`), set the `qio_cache_enable` flag using the `vxtunefs` command after you mount the file system.

To enable the `qio_cache_enable` flag for a file system

- ◆ Use the `vxtunefs` command as follows:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /db02
```

where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to “1” enables Cached Quick I/O. This command enables caching for all the Quick I/O files on this file system.

To disable the flag on the same file system

- ◆ Use the `vxtunefs` command as follows:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /db02
```

where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to “0” disables Cached Quick I/O. This command disables caching for all the Quick I/O files on this file system.

Making Cached Quick I/O settings persistent across reboots and mounts

You can make the Cached Quick I/O system setting persistent across reboots and mounts by adding a file system entry in the `/etc/vx/tunefstab` file.

Note: The `tunefstab` file is a user-created file. For information on how to create the file and add tuning parameters, see the `tunefstab (4)` manual page.

To enable a file system after rebooting

- ◆ Put the file system in the `/etc/vx/tunefstab` file and set the flag entry:

```
/dev/vx/dsk/dgname/volname qio_cache_enable=1
```

where:

- `/dev/vx/dsk/dgname/volname` is the name of a block device
- `dgname` is the name of the disk group
- `volname` is the name of the volume

For example:

```
/dev/vx/dsk/PRODDg/db01 qio_cache_enable=1  
/dev/vx/dsk/PRODDg/db02 qio_cache_enable=1
```

where `/dev/vx/dsk/PRODDg/db01` is the block device on which the file system resides.

The `tunefstab (4)` manual pages contain information on how to add tuning parameters.

See the `tunefstab (4)` manual page.

Note: `vxtunefs` can specify a mount point or a block device; `tunefstab` must always specify a block device only.

Using `vxtunefs` to obtain tuning information

Check the setting of the `qio_cache_enable` flag for each file system using the `vxtunefs` command.

To obtain information on only the `qio_cache_enable` flag setting

- ◆ Use the `grep` command with `vxtunefs`:

```
# /opt/VRTS/bin/vxtunefs /mount_point | grep qio_cache_enable
```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01 | grep qio_cache_enable
```

where `/db01` is the name of the file system. This command displays only the `qio_cache_enable` setting as follows:

```
qio_cache_enable = 0
```

You can also use the `vxtunefs` command to obtain a more complete list of I/O characteristics and tuning statistics.

See the `vxtunefs (1)` manual page.

To obtain information on all `vxtunefs` system parameters

- ◆ Use the `vxtunefs` command without `grep`:

```
# /opt/VRTS/bin/vxtunefs /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01
```

The `vxtunefs` command displays output similar to the following:

```
Filesystem i/o parameters for /db01
read_pref_io = 2097152
read_nstream = 1
read_unit_io = 2097152
write_pref_io = 2097152
write_nstream = 1
write_unit_io = 2097152
pref_strength = 10
buf_breakup_size = 2097152
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 1
write_throttle = 0
max_diskq = 33554432
initial_extent_size = 8
```

```
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 222425088
fcl_keeptime = 0
fcl_winterval = 3600
fcl_ointerval = 600
oltp_load = 0
```

The `vxtunefs(1)` manual pages contain a complete description of `vxtunefs` parameters and the tuning instructions.

See the `vxtunefs(1)` manual page.

Determining candidates for Cached Quick I/O

Determining which files can benefit from Cached Quick I/O is an iterative process that varies with each application. For this reason, you may need to complete the following steps more than once to determine the best possible candidates for Cached Quick I/O.

Before determining candidate files for Quick I/O, make sure the following conditions have been met:

- | | |
|---------------|--|
| Prerequisites | ■ You must enable Cached Quick I/O for the file systems.
See “Enabling Cached Quick I/O on a file system” on page 90. |
| Usage notes | ■ See the <code>qiostat (1M)</code> manual page for more information. |

Collecting I/O statistics

Once you have enabled Cached Quick I/O on a file system, you need to collect statistics to determine and designate the files that can best take advantage of its benefits.

To collect statistics needed to determine files that benefit from Cached Quick I/O

- 1 Reset the `qiostat` counters by entering:

```
$ /opt/VRTS/bin/qiostat -r /mount_point/filenames
```

- 2 Run the database under full normal load and through a complete cycle (24 to 48 hours in most cases) to determine your system I/O patterns and database traffic in different usage categories (for example, OLTP, reports, and backups) at different times of the day.

- 3 While the database is running, run `qiostat -l` to report the caching statistics as follows:

```
$ /opt/VRTS/bin/qiostat -l /mount_point/filenames
```

or, use the `-i` option to see statistic reports at specified intervals:

```
$ /opt/VRTS/bin/qiostat -i n /mount_point/filenames
```

where `n` is time in seconds

For example:

To collect I/O statistics from all database device files on file system `/db01`:

```
$ /opt/VRTS/bin/qiostat -l /db01/*.dbf
```

About I/O statistics

The output of the `qiostat` command is the primary source of information to use in deciding whether to enable or disable Cached Quick I/O on specific files. Statistics are printed in two lines per object.

The second line of information is defined as follows:

- `CREADS` is the number of reads from the VxFS cache (or total number of reads to Quick I/O files with cache advisory on)
- `PREADS` is the number of reads going to the disk for Quick I/O files with the cache advisory on
- `HIT RATIO` is displayed as a percentage and is the number of `CREADS` minus the number of `PREADS` times 100 divided by the total number of `CREADS`. The formula looks like this:

$$(\text{CREADS} - \text{PREADS}) * 100 / \text{CREADS}$$

The `qiostat -l` command output looks similar to the following:

```
/db01/sysprocs.dbf 17128 9634 68509 38536 24.8 0.4
17124 15728 8.2

/db1/master.dbf 6 1 21 4 10.0 0.0
6 6 0.0

/db01/user.dbf 62552 38498 250213 153992 21.9 0.4
62567 49060 21.6
```

Analyze the output to find out where the cache-hit ratio is above a given threshold. A cache-hit ratio above 20 percent on a file for a given application may be sufficient to justify caching on that file. For systems with larger loads, the acceptable ratio may be 30 percent or above. Cache-hit-ratio thresholds vary according to the database type and load.

Using the sample output above as an example, the file `/db01/master.dbf` does not benefit from the caching because the cache-hit ratio is zero. In addition, the file receives very little I/O during the sampling duration.

However, the file `/db01/user.dbf` has a cache-hit ratio of 21.6 percent. If you have determined that, for your system and load, this figure is above the acceptable threshold, it means the database can benefit from caching. Also, study the numbers reported for the read and write operations. When you compare the number of reads and writes for the `/db01/user.dbf` file, you see that the number of reads is roughly twice the number of writes. You can achieve the greatest performance gains with Cached Quick I/O when using it for files that have higher read than write activity.

Based on these two factors, `/db01/user.dbf` is a prime candidate for Cached Quick I/O.

See “[Enabling and disabling Cached Quick I/O for individual files](#)” on page 96.

Effects of read-aheads on I/O statistics

The number of `CREADS` in the `qiostat` output is the total number of reads performed, including Cached Quick I/O, and the number of `PREADS` is the number of physical reads. The difference between `CREADS` and `PREADS` (`CREADS - PREADS`) is the number of reads satisfied from the data in the file system cache. Thus, you expect that the number of `PREADS` would always be equal to or lower than the number of `CREADS`.

However, the `PREADS` counter also increases when the file system performs read-aheads. These read-aheads occur when the file system detects sequential reads. In isolated cases where cache hits are extremely low, the output from `qiostat` could show that the number of `CREADS` is lower than the number of `PREADS`.

The cache-hit ratio calculated against these `CREAD/PREAD` values is misleading when used to determine whether Cached Quick I/O should be enabled or disabled.

Under these circumstances, you can make a more accurate decision based on a collective set of statistics by gathering multiple sets of data points. Consequently, you might want to enable Cached Quick I/O for all the device files used by a given database, even if just one of the files exhibited a high cache-hit ratio.

Other tools for analysis

While the output of the `qiostat` command is the primary source of information to use in deciding whether to enable Cached Quick I/O on specific files, we also recommend using other tools in conjunction with `qiostat`. For example, benchmarking software that measures database throughput is also helpful. If a benchmark test in which Cached Quick I/O was enabled for a certain set of data files resulted in improved performance, you can also use those results as the basis for enabling Cached Quick I/O.

Enabling and disabling Cached Quick I/O for individual files

After using `qiostat` or other analysis tools to determine the appropriate files for Cached Quick I/O, you need to disable Cached Quick I/O for those individual files that do not benefit from caching using the `qioadmin` command.

- | | |
|---------------|--|
| Prerequisites | ■ Enable Cached Quick I/O for the file system before enabling or disabling Cached Quick I/O at the individual file level. |
| Usage notes | ■ You can enable or disable Cached Quick I/O for individual files while the database is online.
■ You should monitor files regularly using <code>qiostat</code> to ensure that a file's cache-hit ratio has not changed enough to reconsider enabling or disabling Cached Quick I/O for the file.
■ Enabling or disabling Cached Quick I/O for an individual file is also referred to as setting the cache advisory on or off.
■ See the <code>qioadmin (1)</code> manual page. |

Setting cache advisories for individual files

You can enable and disable Cached Quick I/O for individual files by changing the cache advisory settings for those files.

To disable Cached Quick I/O for an individual file

- ◆ Use the `qioadmin` command to set the cache advisory to `OFF` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=OFF /mount_point
```

For example, to disable Cached Quick I/O for the file `/db01/master.dbf`, set the cache advisory to `OFF`:

```
$ /opt/VRTS/bin/qioadmin -S master.dbf=OFF /db01
```

To enable Cached Quick I/O for an individual file

- ◆ Use the `qioadmin` command to set the cache advisory to `ON` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=ON /mount_point
```

For example, running `qiostat` shows the cache hit ratio for the file `/db01/master.dbf` reaches a level that would benefit from caching. To enable Cached Quick I/O for the file `/db01/master.dbf`, set the cache advisory to `ON`:

```
$ /opt/VRTS/bin/qioadmin -S master/dbf=ON /db01
```

Making individual file settings for Cached Quick I/O persistent

You can make the enable or disable individual file settings for Cached Quick I/O persistent across reboots and mounts by adding cache advisory entries in the `/etc/vx/qioadmin` file.

Cache advisories set using the `qioadmin` command are stored as extended attributes of the file in the inode. These settings persist across file system remounts and system reboots, but these attributes are not backed up by the usual backup methods, so they cannot be restored. Therefore, always be sure to reset cache advisories after each file restore. This is not necessary if you maintain the cache advisories for Quick I/O files in the `/etc/vx/qioadmin` file.

To enable or disable individual file settings for Cached Quick I/O automatically after a reboot or mount

- ◆ Add cache advisory entries in the `/etc/vx/qioadmin` file as follows:

```
device=/dev/vx/dsk/<diskgroup>/<volume>  
  
filename,OFF  
  
filename,OFF  
  
filename,OFF  
  
filename,ON
```

For example, to make the Cached Quick I/O settings for individual files in the `/db01` file system persistent, edit the `/etc/vx/qioadmin` file similar to the following:

```
#  
# List of files to cache in /db01 file system  
#  
device=/dev/vx/dsk/PRODDg/db01  
  
user.dbf,ON  
sysprocs.dbf,OFF  
master.dbf,OFF
```

Determining individual file settings for Cached Quick I/O using `qioadmin`

You can determine whether Cached Quick I/O is enabled or disabled for individual files by displaying the file's cache advisory setting using the `qioadmin` command.

Note: To verify caching, always check the setting of the flag `qio_cache_enable` using `vxtunefs`, along with the individual cache advisories for each file.

To display the current cache advisory settings for a file

- ◆ Use the `qioadmin` command with the `-P` option as follows:

```
$ /opt/VRTS/bin/qioadmin -P filename /mount_point
```

For example, to display the current cache advisory setting for the file `sysprocs.dbf` in the `/db01` file system:

```
$ /opt/VRTS/bin/qioadmin -P sysprocs.dbf /db01  
sysprocs.dbf,OFF
```


Improving database performance with Veritas Concurrent I/O

This chapter includes the following topics:

- [About Concurrent I/O](#)
- [Enabling and disabling Concurrent I/O](#)

About Concurrent I/O

Veritas Concurrent I/O improves the performance of regular files on a VxFS file system without the need for extending namespaces and presenting the files as devices. This simplifies administrative tasks and allows databases, which do not have a sequential read/write requirement, to access files concurrently. This chapter describes how to use the Concurrent I/O feature.

With DB2 8.2.2 or later, you can use the Veritas Concurrent I/O feature to improve data write performance for DMS tablespaces with FILE containers. Concurrent I/O can also improve data write performance for most SMS tablespaces.

Quick I/O is still an alternative solution for DMS tablespaces.

In some cases (for example, if the system has extra memory), Cached Quick I/O may further enhance performance.

How Concurrent I/O works

Traditionally, UNIX semantics require that read and write operations on a file occur in a serialized order. Because of this, a file system must enforce strict

ordering of overlapping read and write operations. However, databases do not usually require this level of control and implement concurrency control internally, without using a file system for order enforcement.

The Veritas Concurrent I/O feature removes these semantics from the read and write operations for databases and other applications that do not require serialization.

The benefits of using Concurrent I/O are:

- Concurrency between a single writer and multiple readers
- Concurrency among multiple writers
- Minimalization of serialization for extending writes
- All I/Os are direct and do not use file system caching
- I/O requests are sent directly to file systems
- Inode locking is avoided

Enabling and disabling Concurrent I/O

Concurrent I/O is not turned on by default and must be enabled manually. You will also have to manually disable Concurrent I/O if you choose not to use it in the future.

Enabling Concurrent I/O for DB2

Because you do not need to extend name spaces and present the files as devices, you can enable Concurrent I/O on regular files.

For DB2, you can enable an entire file system to use Concurrent I/O or you can enable specific SMS containers to use Concurrent I/O. If you enable a specific SMS container, the rest of the file system will use the regular buffer I/O.

Warning: For DB2, If you use the `-o cio` option with the `mount` command to mount your primary database file systems, the Concurrent I/O settings will not be preserved when using Database FlashSnap commands or the command.

Before enabling Concurrent I/O, review the following:

- Prerequisites
- To use the Concurrent I/O feature, the file system must be a VxFS file system.
 - Make sure the mount point on which you plan to mount the file system exists.
 - Make sure the DBA can access the mount point.

For DB2, */mount_point* is the directory in which you can put data containers of the SMS tablespaces using the Concurrent I/O feature.

Note: This applies to both creating a new tablespace to use Concurrent I/O or enabling an existing tablespace to use Concurrent I/O.

For example for DB2 to mount a file system named `/datavol` on a mount point named `/db2data`:

```
# /usr/sbin/mount -V vxfs -o cio /dev/vx/dsk/db2dg/datavol \  
/db2data
```

To enable Concurrent I/O on a new SMS container using the `namefs -o cio` option

- ◆ Using the `mount` command, mount the directory in which you want to put data containers of the SMS tablespaces using the Concurrent I/O feature.

```
# /usr/sbin/mount -Vt namefs -o cio /path_name /new_mount_point
```

where:

- */path_name* is the directory in which the files that will be using Concurrent I/O reside
- */new_mount_point* is the new target directory that will use the Concurrent I/O feature

The following is an example of mounting a directory (where the new SMS containers are located) to use Concurrent I/O.

To mount an SMS container named `/container1` on a mount point named `/mysms`:

```
# /usr/sbin/mount -Vt namefs -o cio /datavol/mysms/container1 /mysms
```

To enable Concurrent I/O on an existing SMS container using the `namefs -o cio` option

- 1 Stop the DB2 instance using the `db2stop` command.
- 2 Make the directory that will have Concurrent I/O turned on available using the `mv` command.

```
# mv /mydb/mysmsdir /mydb/mysmsdir2
```

- 3 Remount `/mydb/mysmsdir2` on `/mydb/mysmsdir` using the `mount` command with the `-o cio` option.

```
# mount -Vt namefs -o cio /mydb/mysmsdir2 /mydb/mysmsdir
```

- 4 Start the DB2 instance using the `db2start` command.

```
# db2stop  
# mv /mydb/mysmsdir /mydb/mysmsdir2  
# mount -Vt namefs -o cio /mydb/mysmsdir2 /mydb/mysmsdir  
# db2start
```

This example shows how to mount a directory for an existing SMS container to use Concurrent I/O.

To enable Concurrent I/O on a DB2 tablespace when creating the tablespace

- 1 Use the `db2 -v "create regular tablespace..."` command with the `no file system caching` option.
- 2 Set all other parameters according to your system requirements.

To enable Concurrent I/O on an existing DB2 tablespace

- ◆ Use the DB2 `no file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name no file system caching"
```

where *tablespace_name* is the name of the tablespace for which you are enabling Concurrent I/O.

To verify that Concurrent I/O has been set for a particular DB2 tablespace

- 1 Use the DB2 `get snapshot` option to check for Concurrent I/O.

```
# db2 -v "get snapshot for tablespaces on dbname"
```

where *dbname* is the database name.

- 2 Find the tablespace you want to check and look for the `File system caching` attribute. If you see `File system caching = No`, then Concurrent I/O is enabled.

Disabling Concurrent I/O for DB2

If you need to disable Concurrent I/O, use the DB2 `file system caching` option.

Note: If you used the `namefs -o cio` option with the `mount` command to mount a directory to enable Concurrent I/O, make sure you remount without that option as well. Also, if you follow the directions for enabling Concurrent I/O on an existing SMS container, rename the directory back to the original name.

To disable Concurrent I/O on a DB2 tablespace

- ◆ Use the DB2 `file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name file system caching"
```

where *tablespace_name* is the name of the tablespace for which you are disabling Concurrent I/O.

Enabling Concurrent I/O for Sybase

Because you do not need to extend name spaces and present the files as devices, you can enable Concurrent I/O on regular files.

Before enabling Concurrent I/O, review the following:

- | | |
|---------------|---|
| Prerequisites | <ul style="list-style-type: none">■ To use the Concurrent I/O feature, the file system must be a VxFS file system.■ Make sure the mount point on which you plan to mount the file system exists.■ Make sure the DBA can access the mount point. |
|---------------|---|

To enable Concurrent I/O on a file system using mount with the -o cio option

- ◆ Mount the file system using the `mount` command as follows:

```
# /usr/sbin/mount -V vxfs -o cio special /mount_point
```

where:

- *special* is a block special device.
- */mount_point* is the directory where the file system will be mounted.

For example for Sybase, to mount a file system named */datavol* on a mount point named */sybasedata*:

```
# /usr/sbin/mount -V vxfs -o cio /dev/vx/dsk/sybasedg/datavol \  
/sybasedata
```

The following is an example of mounting a directory (where the new SMS containers are located) to use Concurrent I/O.

To mount an SMS container named */container1* on a mount point named */mysms*:

```
# /usr/sbin/mount -Vt namefs -o cio /datavol/mysms/container1 /mysms
```

Disabling Concurrent I/O for Sybase

If you need to disable Concurrent I/O, unmount the VxFS file system and mount it again without the `-o cio` mount option.

To disable Concurrent I/O on a file system using the mount command

- 1 Shutdown the DB2 instance.
- 2 Unmount the file system using the `umount` command.
- 3 Mount the file system again using the `mount` command without using the `-o cio` option.

Storage Foundation Thin Storage optimization

- [Chapter 8. About SF Thin Storage optimization solutions](#)
- [Chapter 9. Migrating data from thick storage to thin storage](#)
- [Chapter 10. Using SF Thin Reclamation](#)

About SF Thin Storage optimization solutions

This chapter includes the following topics:

- [About SF solutions for thin optimization](#)
- [About Thin Storage](#)
- [About Thin Provisioning](#)
- [About SF Thin Reclamation feature](#)
- [About SmartMove](#)

About SF solutions for thin optimization

Array-based options like Thin Storage and Thin Provisioning help storage administrators to meet the challenges in managing storage, such as provisioning storage, migrating data for storage utilization, and maintaining storage utilization. Storage Foundation provides several features that work together with the array functionality to solve these challenges.

This topic describes the following solutions:

- Provisioning Storage Foundation with SmartMove
- Migrating data from thick storage to thin storage
- Reclaiming space on a thin storage device with Thin Reclamation capabilities

About Thin Storage

Thin Storage is an array vendor solution for allocating storage to applications only when the storage is truly needed, from a pool of free storage. Thin Storage attempts to solve the problem of under utilization of available array capacity.

Thin Storage Reclamation-capable arrays and LUNs allow the administrators to release once-used storage to the pool of free storage. Storage is allocated from the free pool when files are created and written to in the file system. However, this storage is not released to the free pool when files get deleted; the administrator must perform the operation of reclaiming this storage for the free pool.

Veritas File System supports reclamation of the free blocks in the file system on Veritas Volume Manager-backed file systems. The operation of reclamation can be done on a disk group, LUN, enclosure, full file system, or part of a file system using the `vxdisk` and `fsadm` commands, and the `vxfs_ts_reclaim` API.

About Thin Provisioning

Thin Provisioning is a storage array feature that optimizes storage use by automating storage provisioning. Administrators do not have to estimate how much storage an application requires. Instead, Thin Provisioning lets administrators provision large thin or thin reclaim capable LUNs to a host. Physical storage capacity is allocated from a thin pool to the thin/thin reclaim capable LUNS only after application I/O writes.

About SF Thin Reclamation feature

You can use the Thin Reclamation feature in the following ways:

- Space is reclaimed automatically when a volume is deleted. Because it is asynchronous, you may not see the reclaimed space immediately.
- You can trigger reclamation for a disk, disk group, or enclosure.
- You can trigger reclamation for a VxFS file system.

About SmartMove

With Storage Foundation's SmartMove feature, Veritas File System (VxFS) lets Veritas Volume Manager (VxVM) know which blocks have data. VxVM, which is the copy engine for migration, copies only the used blocks and avoids copying unused blocks. This behavior helps optimize the thin storage utilization.

Using SmartMove with Thin Provisioning

This section describes how to use SmartMove with Thin Provisioning that improves the synchronization performance and uses thin storage efficiently.

To use SmartMove with Thin Provisioning

- 1 Mount the volume as the VxFS file system type. For example:

```
# mount -V vxfs /dev/vx/dsk/oradg/oravol1 /oravol1
```

- 2 Run the following command:

```
# sync
```

- 3 Mirror the volume. For example:

```
# vxassist -g oradg mirror oravol1
```


Migrating data from thick storage to thin storage

This chapter includes the following topics:

- [About using SmartMove to migrate to Thin Storage](#)
- [Setting up SmartMove](#)
- [Migrating to thin provisioning](#)

About using SmartMove to migrate to Thin Storage

If you have existing data on a thick LUN, the SmartMove feature enables you to migrate that data to a thin LUN, and reclaim the unused space. The SmartMove feature leverages the Veritas File System (VxFS) information about which blocks in a Veritas Volume Manager (VxVM) volume contain data. Therefore, the migration functionality is available only when a VxVM volume is on a mounted VxFS file system.

By default, the SmartMove feature is enabled for all volumes. Setting up the SmartMove feature is only required if you have disabled the feature previously.

After SmartMove is enabled, migrate the data to the thin LUN, following the recommended procedure.

Setting up SmartMove

This section describes how to set up SmartMove.

Displaying the SmartMove configuration

This section describes how to display the SmartMove configuration.

To display the SmartMove value

- ◆ To display the current and default SmartMove values, type the following command:

```
# vxdefault list
KEYWORD                                CURRENT-VALUE  DEFAULT-VALUE
usefssmartmove                          all            all
...
```

Changing the SmartMove configuration

SmartMove has three different values where SmartMove can be applied or not. The three values are:

Value	Meaning
none	Do not use SmartMove at all.
thinonly	Use SmartMove for thin aware LUNs only.
all	Use SmartMove for all operations. This is the default value.

To set the SmartMove value

- ◆ To set the SmartMove to apply to all volumes, type the following command:

```
# vxdefault set usefssmartmove value
```

where *value* is either `none`, `thinonly`, or `all`.

Migrating to thin provisioning

The SmartMove™ feature enables migration from traditional LUNs to thinly provisioned LUNs, removing unused space in the process.

See the *Veritas Volume Manager Administrator's Guide* for more information on SmartMove and Thin Provisioning.

To migrate to thin provisioning

- 1 Check if the SmartMove Feature is enabled.

See “[Displaying the SmartMove configuration](#)” on page 114.

See “[Changing the SmartMove configuration](#)” on page 114.

- 2 Add the new, thin LUNs to the existing disk group. Enter the following commands:

```
# vxdisksetup -i da_name
# vxdg -g datadg adddisk da_name
```

where *da_name* is the disk access name in VxVM.

- 3 To identify LUNs with the `thinonly` or `thinrdlm` attributes, enter:

```
# vxdisk -o thin list
```

- 4 Add the new, thin LUNs as a new plex to the volume.

NOTE: The VxFS file system must be mounted to get the benefits of the SmartMove feature.

The following methods are available to add the LUNs:

- Use the default settings for the `vxassist` command:

```
# vxassist -g datadg mirror datavol da_name
```

- Specify the `vxassist` command options for faster completion. The `-b` option copies blocks in the background. The following command has more I/O affect:

```
# vxassist -b -oiosize=1m -t thinmig -g datadg mirror \
  datavol da_name
# vxtask monitor thinmig
```

- Specify the `vxassist` command options for minimal effect. The following command takes longer to complete:

```
# vxassist -oslow -g datadg mirror datavol da_name
```

- 5 Optionally, test the performance of the new LUNs before removing the old LUNs.

To test the performance, use the following steps:

- Determine which plex corresponds to the thin LUNs:

```
# vxprint -g datadg
```

TY NAME	ASSOC	KSTATE	LENGTH	PLOFFS	STATE	TUTILO	PUTILO
dg datadg	datadg	-	-	-	-	-	-
dm THINARRAY0_02	THINARRAY0_02	-	83886080	-	-	-	-
dm STDARRAY1_01	STDARRAY1_01	-	41943040	-	-OHOTUSE	-	-
v datavol	fsgen	ENABLED	41943040	-	ACTIVE	-	-
pl datavol-01	datavol	ENABLED	41943040	-	ACTIVE	-	-
sd STDARRAY1_01-01	datavol-01	ENABLED	41943040	0	-	-	-
pl datavol-02	datavol	ENABLED	41943040	-	ACTIVE	-	-
sd THINARRAY0_02-01	datavol-02	ENABLED	41943040	0	-	-	-

The above output indicates that the thin LUN corresponds to plex datavol-02.

- Direct all reads to come from those LUNs:

```
# vxvol -g datadg rdpol prefer datavol datavol-02
```

6 Remove the original non-thin LUNs.

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

```
# vxassist -g datadg remove mirror datavol \!STDARRAY1_01  
# vxdg -g datadg rmdisk STDARRAY1_01  
# vxdisk rm STDARRAY1_01
```

7 Grow the file system and volume to use all of the larger thin LUN:

```
# vxresize -g datadg -x datavol 40g da_name
```

Using SF Thin Reclamation

This chapter includes the following topics:

- [Reclamation of storage on thin reclamation arrays](#)
- [Monitoring Thin Reclamation using the `vxtask` command](#)

Reclamation of storage on thin reclamation arrays

Storage Foundation enables reclamation of storage on thin reclamation arrays.

See [“How reclamation on a deleted volume works”](#) on page 118.

The thin reclamation feature is supported only for LUNs that have the `thinrcldm` attribute. VxVM automatically discovers LUNs that support Thin Reclamation from thin capable storage arrays. You can list devices that are known to have the `thinonly` or `thinrcldm` attributes on the host.

See [“Identifying thin and thin reclamation LUNs”](#) on page 117.

Identifying thin and thin reclamation LUNs

You can only perform Thin Reclamation on LUNs which have the `thinrcldm` attribute. VxVM automatically discovers LUNs that support Thin Reclamation from capable storage arrays. To identify devices that are known to have the `thinonly` or `thinrcldm` attributes on a host, use the `vxdisk -o thin list` command.

To identify LUNs

- ◆ To identify LUNs that are `thin` or `thinreclm` type the following command:

```
# vxdisk -o thin list
DEVICE          SIZE (mb)    PHYS_ALLOC (mb)  GROUP    TYPE
hitachi_usp0_065a 10000        84               -        thinreclm
hitachi_usp0_065b 10000        110              -        thinreclm
hitachi_usp0_065c 10000        74               -        thinreclm
hitachi_usp0_065d 10000        50               -        thinreclm
.
.
.
hitachi_usp0_0660 10000        672              thindg   thinreclm
```

In the above output, the `SIZE` column shows the size of the disk. The `PHYS_ALLOC` column shows the physical allocation on the array side. The `TYPE` indicates that the array supports thin reclamation.

How reclamation on a deleted volume works

Storage that is no longer in use, needs to be reclaimed by the array. The process of reclaiming storage on an array can be intense on the array. To avoid any effect on regular I/O's to the array, the reclaim operation is made asynchronous. When a volume is deleted the space previously used by the volume is tracked for later asynchronous reclamation. This asynchronous reclamation is handled by `vxrelocd` (or recovery) daemon.

By default, the `vxrelocd` daemon runs everyday at 22:10 hours and reclaims storage on the deleted volume that are one day old.

To perform the reclaim operation during less critical time of the system, control the time of the reclaim operation by using the following tunables:

<code>reclaim_on_delete_wait_period</code>	<p>The storage space that is used by the deleted volume is reclaimed after <code>reclaim_on_delete_wait_period</code> days. The value of the tunable can be anything between -1 to 367.</p> <p>The default is set to 1, which means the volume is deleted the next day. The storage is reclaimed immediately if the value is -1. The storage space is not reclaimed automatically, if the value is greater than 366. It can only be reclaimed manually using <code>vxdisk reclaim</code> command.</p>
<code>reclaim_on_delete_start_time</code>	<p>This tunable specifies the time of the day that the reclaim on the deleted volume is performed.</p> <p>The default time is set to 22:10. This value can be changed to any time of the day.</p>

You can change the tunables using the `vxdefault` command.

Thin Reclamation of a disk, a disk group, or an enclosure

Use the `vxdisk reclaim` command to trigger online Thin Reclamation on one or more disks, disk groups, or enclosures. By default, the `vxdisk reclaim` command performs Thin Reclamation on the disks where the VxVM volume is on a “mounted” VxFS file system. The reclamation skips disks that do not have a VxFS file system mounted.

Use the `-o full` option of the `vxdisk reclaim` command to also reclaim disk space in unmarked space on the disks.

You can only perform Thin Reclamation on LUNS which have the `thinreclm` attribute.

See “[Identifying thin and thin reclamation LUNs](#)” on page 117.

Example of reclamation for disks. The following example triggers reclamation on LUNs `disk1` and `disk2`:

```
# vxdisk reclaim disk1 disk2
```

In the above example, suppose the `disk1` contains a VxVM volume `vol1` with a VxFS file system. If the VxFS file system is not mounted, the command skips reclamation for `disk1`.

To reclaim space on `disk1`, use the following command:

```
# vxdisk -o full reclaim disk1
```

The above command reclaims unused space on `disk1` that is outside of the `vol1`. The reclamation skips the `vol1` volume, since the VxFS file system is not mounted, but it scans the rest of the disk for unused space.

Example of reclamation for disk groups. The following example triggers reclamation on the disk group `oradg`:

```
# vxdisk reclaim oradg
```

Example of reclamation for an enclosure. The following example triggers reclamation on the enclosure=`EMC_CLARiiON0`:

```
# vxdisk reclaim EMC_CLARiiON0
```

You can also trigger Thin Reclamation on a VxFS file system.

Thin Reclamation takes considerable amount of time when you reclaim thin storage on a large number of LUNs or an enclosure or disk group.

See [“Monitoring Thin Reclamation using the `vxtask` command”](#) on page 121.

Thin Reclamation of a file system

Veritas File System (VxFS) supports reclamation of free storage on a Thin Storage LUN. Free storage is reclaimed using the `fsadm` command or the `vxfs_ts_reclaim` API. You can perform the default reclamation or aggressive reclamation. If you used a file system for a long time and must perform reclamation on the file system, Symantec recommends that you run aggressive reclamation. Aggressive reclamation compacts the allocated blocks, which creates larger free blocks that can potentially be reclaimed.

See the `fsadm_vxfs(1M)` and `vxfs_ts_reclaim(3)` manual pages.

Thin Reclamation is only supported on file systems mounted on a VxVM volume.

The following example performs aggressive reclamation of free storage to the Thin Storage LUN on a VxFS file system mounted at `/mnt1`:

```
# /opt/VRTS/bin/fsadm -R /mnt1
```

Veritas File System also supports reclamation of a portion of the file system using the `vxfs_ts_reclaim()` API.

See the *Veritas File System Programmer's Reference Guide*.

Note: Thin Reclamation is a slow process and may take several hours to complete, depending on the file system size. Thin Reclamation is not guaranteed to reclaim 100% of the free space.

You can track the progress of the Thin Reclamation process by using the `vxtask list` command when using the Veritas Volume Manager (VxVM) command `vxdisk reclaim`.

See the `vxtask(1M)` and `vxdisk(1M)` manual pages.

You can administer Thin Reclamation using VxVM commands.

See the *Veritas Volume Manager Administrator's Guide*.

Triggering space reclamation

This section describes how to trigger space reclamation.

To trigger space reclamation

- 1 Ensure you mounted the VxFS file system.

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

If you need to mount the VxFS file system, see the `mount_vxfs(1M)` manual page.

- 2 Use the `fsadm` command to trigger space reclamation:

```
# /opt/VRTS/bin/fsadm -V vxfs -R /<VxFS_mount_point>
```

where `<VxFS_mount_point>` is the name of the VxFS file system mount point.

Note: If the VxFS file system is not mounted you will receive an error message. For example: `Disk 3pardata0_110 : Skipped. No VxFS file system found.`

See the *Veritas File System Administrator's Guide* for more information on how to trigger Thin Reclamation on a VxFS file system.

Monitoring Thin Reclamation using the `vxtask` command

This section describes how to monitor thin reclamation using the `vxtask` command.

To monitor thin reclamation

- 1 To initiate thin reclamation, use the following command:

```
# vxdisk reclaim diskgroup
```

For example:

```
# vxdisk reclaim dg100
```

- 2 To monitor the reclamation status, run the following command in another session:

```
# vxtask list
```

```
TASKID PTID TYPE/STATE PCT PROGRESS  
171 RECLAIM/R 00.00% 0/41875931136/0 RECLAIM vol100 dg100
```

The `vxdisk reclaim diskgroup` command runs in another session while you run the `vxtask list` command.

Making point-in-time copies

- [Chapter 11. Understanding point-in-time copy methods](#)
- [Chapter 12. Setting up volumes for instant snapshots](#)
- [Chapter 13. Online database backup](#)
- [Chapter 14. Off-host cluster file system backup](#)
- [Chapter 15. Decision support](#)
- [Chapter 16. Database recovery](#)
- [Chapter 17. Administering volume snapshots](#)
- [Chapter 18. Administering snapshot file systems](#)
- [Chapter 19. Administering Storage Checkpoints](#)
- [Chapter 20. Administering FileSnaps](#)
- [Chapter 21. Backing up and restoring with Netbackup in an SFHA environment](#)

Understanding point-in-time copy methods

This chapter includes the following topics:

- [About point-in-time copies](#)
- [About point-in-time copy technology](#)
- [Point-in-time copy use cases](#)

About point-in-time copies

Two trends dominate the evolution of digital data used to conduct and manage business. First, more and more data must be continuously available for 24x7 transaction processing, decision making, intellectual property creation, and so forth. Second, as digital data assumes a larger role in the conduct of business, protecting it from loss or destruction becomes increasingly important. These trends run counter to each other. To protect it properly, data must be taken out of service so that it stands still while it is backed up. But taking data out of service means that it's not there when the business needs it. To solve this problem, information technology has developed a set of techniques for freezing instantaneous images of data—for taking snapshots of it—that can be used to make backups, perform analyses, and test software updates, all while production applications continue to operate on the live data.

Point-in-time copies of volumes allow you to capture an image of a database or file system at a selected instant for use in applications such as backups, decision support, reporting, and development testing.

Point-in-time copy solutions may additionally be configured to use off-host processing to remove much of the performance overhead on a production system.

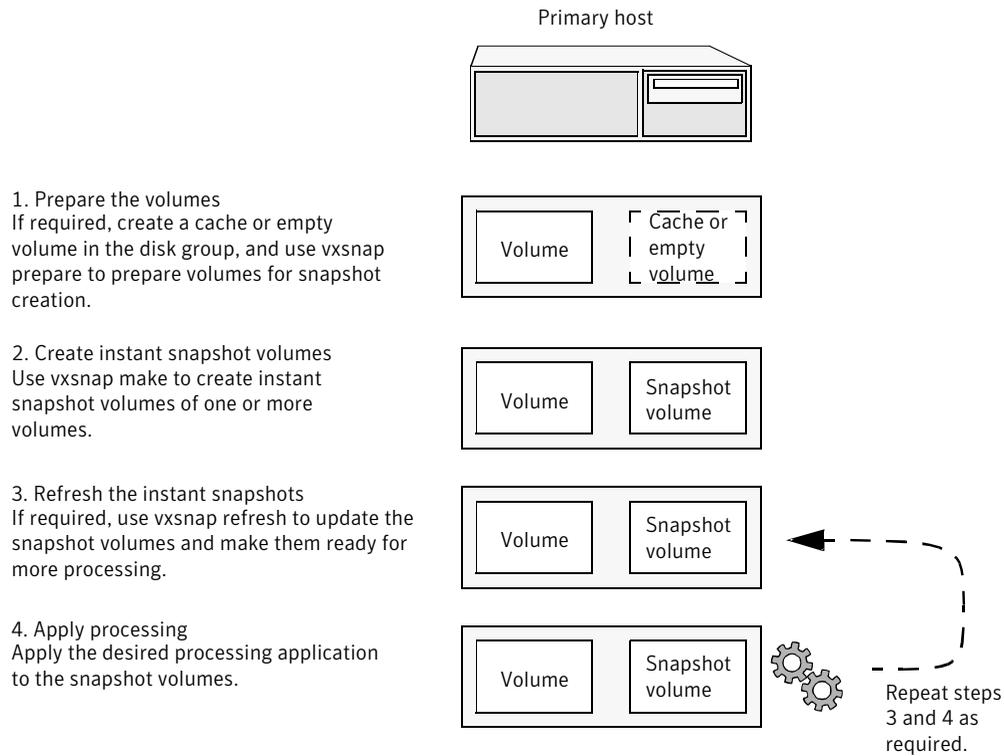
The following types of point-in-time copy solution are considered in this document:

- Primary host solutions where the copy is processed on the same system as the active data.
See “[Implementing point-in time copy solutions on a primary host](#)” on page 126.
- Off-host solutions where the copy is processed on a different system from the active data. If implemented correctly, such solutions have almost no impact on the performance of the primary production system.
See “[Implementing off-host point-in-time copy solutions](#)” on page 127.

Implementing point-in time copy solutions on a primary host

Figure 11-1 illustrates the steps that are needed to set up the processing solution on the primary host.

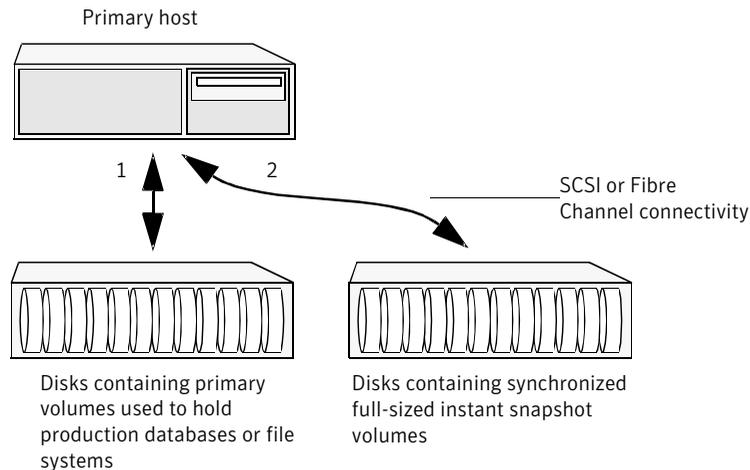
Figure 11-1 Using snapshots and FastResync to implement point-in-time copy solutions on a primary host



Note: The Disk Group Split/Join functionality is not used. As all processing takes place in the same disk group, synchronization of the contents of the snapshots from the original volumes is not usually required unless you want to prevent disk contention. Snapshot creation and updating are practically instantaneous.

Figure 11-2 shows the suggested arrangement for implementing solutions where the primary host is used and disk contention is to be avoided.

Figure 11-2 Example point-in-time copy solution on a primary host



In this setup, it is recommended that separate paths (shown as 1 and 2) from separate controllers be configured to the disks containing the primary volumes and the snapshot volumes. This avoids contention for disk access, but the primary host's CPU, memory and I/O resources are more heavily utilized when the processing application is run.

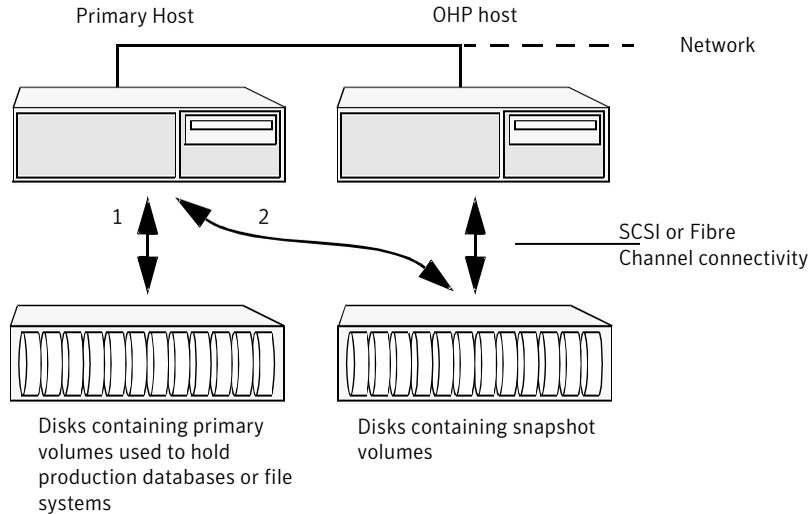
Note: For space-optimized or unsynchronized full-sized instant snapshots, it is not possible to isolate the I/O pathways in this way. This is because such snapshots only contain the contents of changed regions from the original volume. If applications access data that remains in unchanged regions, this is read from the original volume.

Implementing off-host point-in-time copy solutions

Figure 11-3 illustrates that, by accessing snapshot volumes from a lightly loaded host (shown here as the OHP host), CPU- and I/O-intensive operations for online

backup and decision support are prevented from degrading the performance of the primary host that is performing the main production activity (such as running a database).

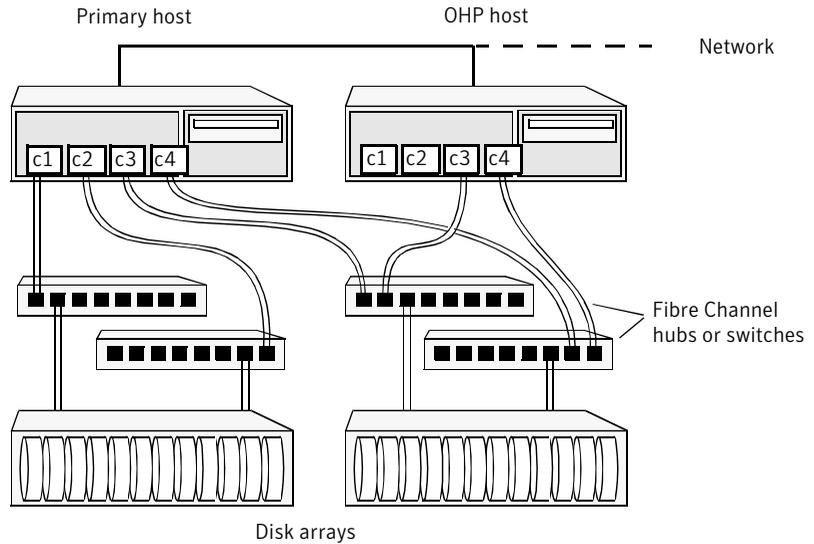
Figure 11-3 Example implementation of an off-host point-in-time copy solution



Also, if you place the snapshot volumes on disks that are attached to host controllers other than those for the disks in the primary volumes, it is possible to avoid contending with the primary host for I/O resources. To implement this, paths 1 and 2 shown in the [Figure 11-3](#) should be connected to different controllers.

[Figure 11-4](#) shows an example of how you might achieve such connectivity using Fibre Channel technology with 4 Fibre Channel controllers in the primary host.

Figure 11-4 Example connectivity for off-host solution using redundant-loop access



This layout uses redundant-loop access to deal with the potential failure of any single component in the path between a system and a disk array.

Note: On some operating systems, controller names may differ from what is shown here.

[Figure 11-5](#) shows how off-host processing might be implemented in a cluster by configuring one of the cluster nodes as the OHP node.

Figure 11-5 Example implementation of an off-host point-in-time copy solution using a cluster node

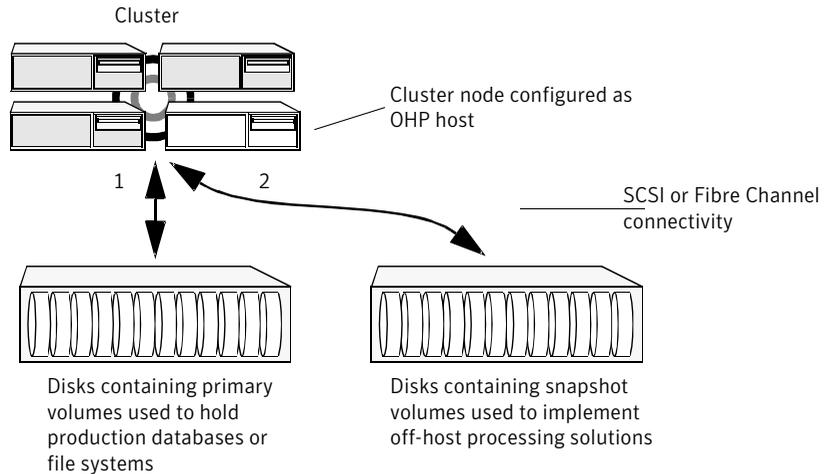
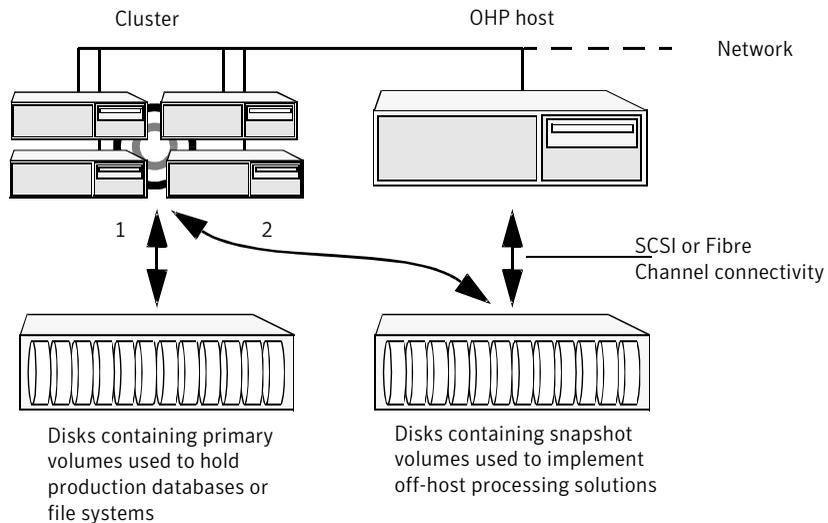


Figure 11-6 shows an alternative arrangement, where the OHP node could be a separate system that has a network connection to the cluster, but which is not a cluster node and is not connected to the cluster's private network.

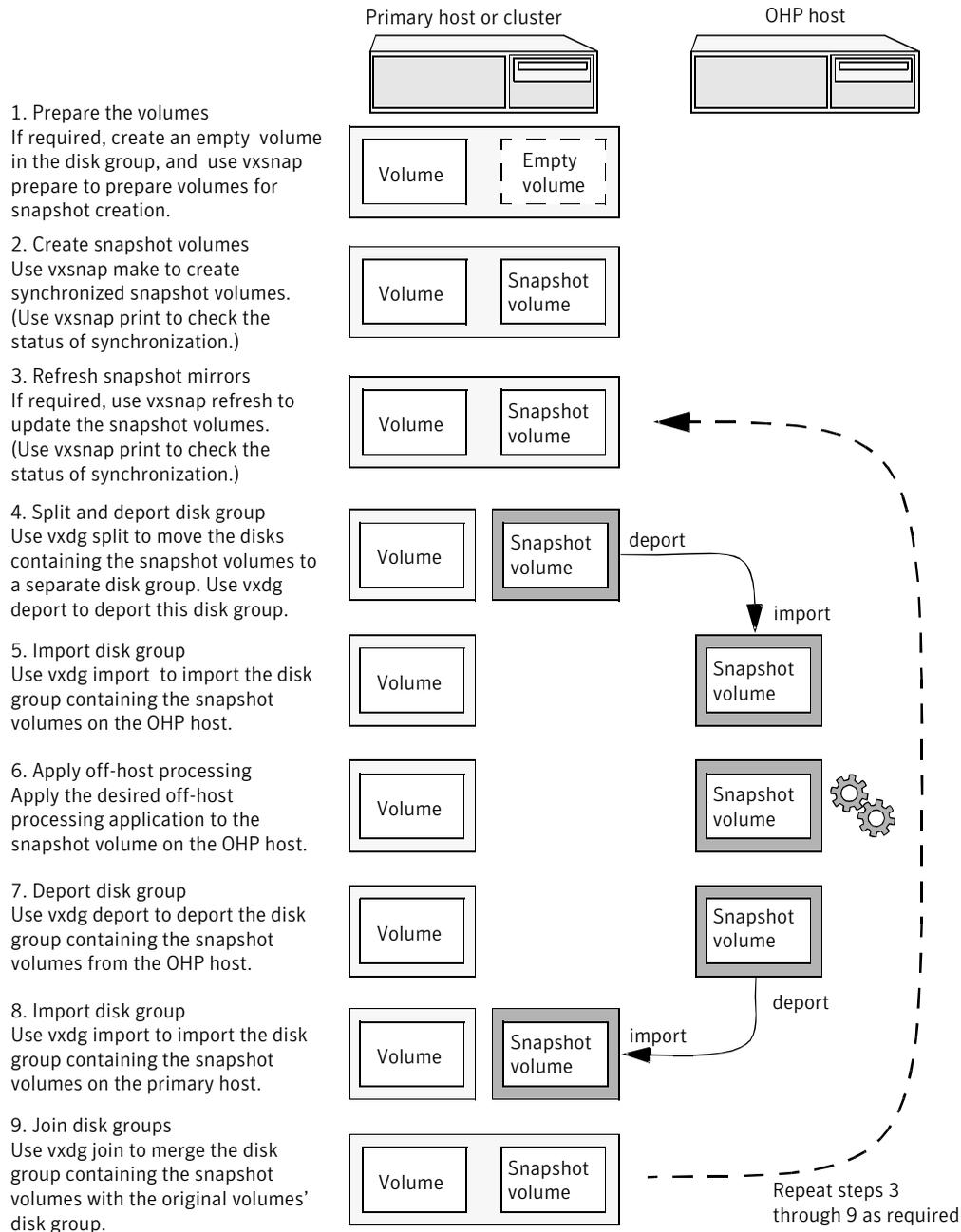
Figure 11-6 Example implementation of an off-host point-in-time copy solution using a separate OHP host



Note: For off-host processing, the example scenarios in this document assume that a separate OHP host is dedicated to the backup or decision support role. For clusters, it may be simpler, and more efficient, to configure an OHP host that is not a member of the cluster.

[Figure 11-7](#) illustrates the steps that are needed to set up the processing solution on the primary host.

Figure 11-7 Implementing off-host processing solutions



Disk Group Split/Join is used to split off snapshot volumes into a separate disk group that is imported on the OHP host.

Note: As the snapshot volumes are to be moved into another disk group and then imported on another host, their contents must first be synchronized with the parent volumes. On reimporting the snapshot volumes, refreshing their contents from the original volume is speeded by using FastResync.

About point-in-time copy technology

This topic introduces the point-in-time copy solutions that you can implement using the Veritas FlashSnap™ technology.

Veritas FlashSnap offers a flexible and efficient means of managing business critical data. It allows you to capture an online image of actively changing data at a given instant: a point-in-time copy. You can perform system backup, upgrade and other maintenance tasks on point-in-time copies while providing continuous availability of your critical data. If required, you can offload processing of the point-in-time copies onto another host to avoid contention for system resources on your production server.

The following kinds of point-in-time copy solution are supported by the FlashSnap license:

- Volume-level solutions are made possible by the persistent FastResync and Disk Group Split/Join features of Veritas Volume Manager. These features are suitable for implementing solutions where the I/O performance of the production server is critical.
See [“Persistent FastResync of volume snapshots”](#) on page 134.
See [“Disk group split/join”](#) on page 136.
- File system-level solutions use the Storage Checkpoint feature of Veritas File System. Storage Checkpoints are suitable for implementing solutions where storage space is critical for:
 - File systems that contain a small number of mostly large files.
 - Application workloads that change a relatively small proportion of file system data blocks (for example, web server content and some databases).
 - Applications where multiple writable copies of a file system are required for testing or versioning.
See [“Storage Checkpoints”](#) on page 137.
- File level snapshots.

The FlashSnap license also supports the Veritas FlashSnap Agent for Symmetrix.

See [“Veritas FlashSnap Agent for Symmetrix”](#) on page 138.

Volume snapshots

A snapshot is a virtual image of the content of a set of data at the instant of creation. Physically, a snapshot may be a full (complete bit-for-bit) copy of the data set, or it may contain only those elements of the data set that have been updated since snapshot creation. The latter are sometimes referred to as allocate-on-first-write snapshots, because space for data elements is added to the snapshot image only when the elements are updated (overwritten) for the first time in the original data set. Storage Foundation allocate-on-first-write snapshots are called space-optimized snapshots.

A full-copy snapshot of a virtual volume requires storage capacity equal to that of the original volume, as well as sufficient I/O bandwidth to copy the entire contents of the original volume to the snapshot within the required time. Space-optimized snapshots, on the other hand, consume storage and I/O bandwidth in proportion to how much data on the original volume is updated during a snapshot’s life. Thus, full-copy snapshots require more storage and I/O bandwidth than space-optimized ones. They are generally more suitable for recovering from storage device failures, and for off-host auxiliary processing where impact on production applications is a concern. They are generally less optimal for protecting against data corruption due to human or application error.

Space-optimized snapshots do not contain complete physical images of the original data objects they represent, and they cannot be taken off-host for auxiliary processing. However, because they consume relatively little storage and I/O bandwidth, they can be taken much more frequently than full-copy snapshots. This makes them well-suited for recovering from data corruption. Space-optimized snapshots naturally tend to grow with age, as more of the data in the original objects changes, so they are inherently better-suited for shorter lifetimes.

Persistent FastResync of volume snapshots

Veritas Volume Manager allows you to take multiple snapshots of your data at the level of a volume. A snapshot volume contains a stable copy of a volume’s data at a given moment in time that you can use for online backup or decision support. If persistent FastResync is enabled on a volume, VxVM uses a FastResync map to keep track of which blocks are updated in the volume and in the snapshot. If the data in one mirror is not updated for some reason, it becomes out-of-date, or stale, with respect to the other mirrors in the volume. The presence of the FastResync map means that only those updates that the mirror has missed need be reapplied to resynchronize it with the volume. A full, and therefore much slower, resynchronization of the mirror from the volume is unnecessary.

When snapshot volumes are reattached to their original volumes, persistent FastResync allows the snapshot data to be quickly refreshed and re-used. Persistent FastResync uses disk storage to ensure that FastResync maps survive both system and cluster crashes. If persistent FastResync is enabled on a volume in a private disk group, incremental resynchronization can take place even if the host is rebooted.

Persistent FastResync can track the association between volumes and their snapshot volumes after they are moved into different disk groups. After the disk groups are rejoined, persistent FastResync allows the snapshot plexes to be quickly resynchronized.

See the *Veritas Volume Manager Administrator's Guide*.

Instant volume snapshots

The traditional type of volume snapshot that was provided in VxVM is of the third-mirror type. This name comes from its original implementation by adding an additional plex to a mirrored volume. After the contents of the third-mirror (or snapshot plex) had been synchronized from the original plexes of the volume, it could be detached as a snapshot volume for use in backup or decision support applications. Enhancements to the snapshot model allowed snapshot volumes to contain more than a single plex, reattachment of a subset of a snapshot volume's plexes, and persistence of FastResync across system reboots or cluster restarts.

Release 4.0 of VxVM introduced instant snapshots, which offer advantages over traditional third-mirror snapshots. The benefits of instant snapshots include immediate availability for use, quick refreshment, and easier configuration and administration. Full-sized instant snapshots are similar to third-mirror snapshots in that they are the same length as the original volume.

Space-optimized instant snapshots require less space than full-sized snapshots by recording changed regions in the original volume to a storage cache. As the original volume is written to, VxVM preserves its data in the cache before the write is committed.

See the *Veritas Volume Manager Administrator's Guide*.

Data integrity in volume snapshots

A volume snapshot represents the data that exists in a volume at a given point in time. As such, VxVM does not have any knowledge of data that is cached by the overlying file system, or by applications such as databases that have files open in the file system. If the `fsген` volume usage type is set on a volume that contains a Veritas File System (VxFS), intent logging of the file system metadata ensures the internal consistency of the file system that is backed up. For other file system

types, depending on the intent logging capabilities of the file system, there may potentially be inconsistencies between in-memory data and the data in the snapshot image.

For databases, a suitable mechanism must additionally be used to ensure the integrity of tablespace data when the volume snapshot is taken. The facility to temporarily suspend file system I/O is provided by most modern database software. The examples provided in this document illustrate how to perform this operation. For ordinary files in a file system, which may be open to a wide variety of different applications, there may be no way to ensure the complete integrity of the file data other than by shutting down the applications and temporarily unmounting the file system. In many cases, it may only be important to ensure the integrity of file data that is not in active use at the time that you take the snapshot.

Choices for snapshot resynchronization

When a snapshot volume is reattached to its original volume within a shared disk group, there are two choices for resynchronizing the data in the volume:

- Resynchronize the snapshot from the original volume—updates the snapshot with data from the primary volume that has changed since the snapshot was taken. The snapshot is then again ready to be taken for the purposes of backup or decision support.
- Resynchronize the original volume from the snapshot—updates the original volume with data from the snapshot volume that has changed since the snapshot was taken. This may be necessary to restore the state of a corrupted database or file system, or to implement upgrades to production software, and is usually much quicker than using alternative approaches such as full restoration from backup media.

Disk group split/join

One or more volumes, such as snapshot volumes, can be split off into a separate disk group and deported. They are then ready for importing on another host that is dedicated to off-host processing. This host need not be a member of a cluster but it must have access to the disks on which the volumes are configured. At a later stage, the disk group can be deported, re-imported, and joined with the original disk group, or with a different disk group.

Note: As space-optimized instant snapshots only record information about changed regions in the original volume, they cannot be moved to a different disk group. They are therefore unsuitable for the off-host processing applications that are described in this document.

The contents of full-sized instant snapshots must be fully synchronized with the unchanged regions in the original volume before such snapshots can be moved into a different disk group and deported from a host.

See the *Veritas Volume Manager Administrator's Guide*.

Storage Checkpoints

A Storage Checkpoint is a persistent image of a file system at a given instance in time. Storage Checkpoints use a copy-on-write technique to reduce I/O overhead by identifying and maintaining only those file system blocks that have changed since a previous Storage Checkpoint was taken. Storage Checkpoints have the following important features:

- Storage Checkpoints persist across system reboots and crashes.
- A Storage Checkpoint can preserve not only file system metadata and the directory hierarchy of the file system, but also user data as it existed when the Storage Checkpoint was taken.
- After creating a Storage Checkpoint of a mounted file system, you can continue to create, remove, and update files on the file system without affecting the image of the Storage Checkpoint.
- Unlike file system snapshots, Storage Checkpoints are writable.
- To minimize disk space usage, Storage Checkpoints use free space in the file system.

Storage Checkpoints and the Storage Rollback feature of Veritas Storage Foundation for Databases enable rapid recovery of databases from logical errors such as database corruption, missing files and dropped table spaces. You can mount successive Storage Checkpoints of a database to locate the error, and then roll back the database to a Storage Checkpoint before the problem occurred.

See [“About database recovery using Storage Checkpoints”](#) on page 195.

Symantec NetBackup for Oracle Advanced BLI Agent uses Storage Checkpoints to enhance the speed of backing up Oracle databases.

See the *Symantec NetBackup for Oracle Advanced BLI Agent System Administrator's Guide*.

For more information about the implementation of Storage Checkpoints, see the *Veritas File System Administrator's Guide*.

Veritas FlashSnap Agent for Symmetrix

The EMC TimeFinder product from EMC is a business continuance solution that allows you to create and use copies of EMC Symmetrix devices while the standard devices remain online and accessible. Business Continuance Volume (BCV) devices contain copies of Symmetrix standard (STD) devices and provide redundancy. You can temporarily detach the BCV mirrors and use the BCVs to perform backups, testing, and other administrative tasks.

Veritas FlashSnap Agent for Symmetrix provides a set of commands that allow you to use the EMC TimeFinder split and restore operations in conjunction with VxFS file systems and VxVM disk groups and volumes that have been created on Symmetrix STD devices.

You can use the commands in Veritas FlashSnap Agent for Symmetrix to:

- Associate a disk group with a BCV disk group, or Symmetrix STD devices in a disk group with identical BCV devices.
- Initiate TimeFinder mirroring for Symmetrix STD devices in a disk group.
- Split Symmetrix STD devices from their BCV devices and create duplicate volumes on the BCV devices. You can use the resulting BCV volumes for administrative tasks such as backups and testing.
- Reattach and resynchronize the STD and BCV devices. The devices can be remirrored from the STD copy or restored from the BCV copy.
- Detach the STD devices from their BCV devices.

Point-in-time copy use cases

The following typical activities are suitable for point-in-time copy solutions implemented using Veritas FlashSnap:

- Data backup—Many enterprises require 24 x 7 data availability. They cannot afford the downtime involved in backing up critical data offline. By taking snapshots of your data, and backing up from these snapshots, your business-critical applications can continue to run without extended downtime or impacted performance.

You can use FlashSnap to implement regular online backups of databases and cluster file system volumes:

See [“About online database backup”](#) on page 155.

See [“About off-host cluster file system backup”](#) on page 167.

- Decision support analysis and reporting—Operations such as decision support analysis and business reporting may not require access to real-time information. You can direct such operations to use a replica database that you have created from snapshots, rather than allow them to compete for access to the primary database. When required, you can quickly resynchronize the database copy with the data in the primary database.
See [“About decision support”](#) on page 177.
- Testing and training—Development or service groups can use snapshots as test data for new applications. Snapshot data provides developers, system testers and QA groups with a realistic basis for testing the robustness, integrity and performance of new applications.
- Database error recovery—Logic errors caused by an administrator or an application program can compromise the integrity of a database. You can recover a database more quickly by restoring the database files by using Storage Checkpoints or a snapshot copy than by full restoration from tape or other backup media.
Use Storage Checkpoints to quickly roll back a database instance to an earlier point in time.
See [“About database recovery using Storage Checkpoints”](#) on page 195.

Setting up volumes for instant snapshots

This chapter includes the following topics:

- [About setting up volumes for instant snapshots](#)
- [Additional preparation activities](#)
- [Preparing a volume for instant snapshot operations](#)
- [Creating a volume for use as a full-sized instant snapshot](#)
- [Creating a shared cache object](#)

About setting up volumes for instant snapshots

This chapter describes how to make volumes ready for instant snapshot creation. These may be volumes that you want to back up, or that you want to use for decision support or reporting.

If a snapshot volume is to be used on the same host, and will not be moved to another host for off-host processing, you can use space-optimized instant snapshots rather than full-sized instant snapshots. Depending on the application, space-optimized snapshots typically require 10% of the disk space that is required for full-sized instant snapshots.

For information on administering instant snapshots and FastResync, see the *Veritas Volume Manager Administrator's Guide*.

[Table 12-1](#) summarizes which volumes require the creation of snapshot mirrors for backup, decision support, and database error recovery.

Table 12-1 Creation of snapshot mirrors

Point-in-time copy application	Create snapshot mirrors for volumes containing...
Online database backup See “ About online database backup ” on page 155.	VxFS file systems for database datafiles to be backed up.
Off-host cluster file system backup See “ About off-host cluster file system backup ” on page 167.	VxFS cluster file systems to be backed up.
Decision support See “ About decision support ” on page 177.	VxFS file systems for database datafiles to be replicated.

Warning: To avoid data inconsistencies, do not use the same snapshot with different point-in-time copy applications. If you require snapshot mirrors for more than one application, configure at least one snapshot mirror that is dedicated to each application.

If the existing volume was created before release 4.0 of VxVM, and it has any attached snapshot plexes, is associated with any snapshot volumes, or has any dedicated DRL logs, follow the procedure given in the section “Upgrading Existing Volumes to Use Version 20 DCOs” in the “Administering Volumes” chapter of the *Veritas Volume Manager Administrator’s Guide*. The procedure given in this section assumes that no snapshot plexes, snapshot volumes, or DRL logs are associated with the volumes.

Additional preparation activities

Depending on the type of snapshots that you want to create, you may need to perform additional preparatory tasks.

When creating a full-sized instant snapshot, you can use one of the following two methods:

- Break off one or more spare plexes from the original volume to form a snapshot volume with the required redundancy. These plexes must be in the `SNAPDONE` state. (You can also break off named plexes of a volume that are in the `ACTIVE` state, but that method is not described here.)

For more information about Administering Volume Snapshots:

See [“About volume snapshots”](#) on page 199.

- Use a separate empty volume that you have prepared in advance.

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

When creating space-optimized instant snapshots that share a cache, you must set up the cache before creating the snapshots.

See [“Creating a shared cache object”](#) on page 151.

If a space-optimized instant snapshot uses a dedicate cache, this can also be set up when the snapshot is created. No additional preparation is required in this case.

Note: The off-host processing solutions in this book require full-sized snapshots.

Preparing a volume for instant snapshot operations

To prepare a volume for instant snapshot operations, a version 20 Data Change Object (DCO) and DCO volume must first be associated with that volume.

To add a version 20 DCO object and DCO volume to an existing volume

- 1 Ensure that the disk group containing the existing volume has been upgraded to at least version 110. Use the following command to check the version of a disk group:

```
# vxprint -l diskgroup | egrep 'version:'
```

To upgrade a disk group, use the following command:

```
# vxdg upgrade diskgroup
```

- 2 Use the following command to add a version 20 DCO and DCO volume to an existing volume:

```
# vxsnap [-g diskgroup] prepare volume [ndcomirs=number] \  
    [regionsize=size] [alloc=storage_attribute[,...]]
```

The `ndcomirs` attribute specifies the number of DCO plexes that are created in the DCO volume. It is recommended that you configure as many DCO plexes as there are data and snapshot plexes in the volume. The DCO plexes are used to set up a DCO volume for any snapshot volume that you subsequently create from the snapshot plexes. For example, specify `ndcomirs=5` for a volume with 3 data plexes and 2 snapshot plexes.

The value of the `regionsize` attribute specifies the size of the tracked regions in the volume. A write to a region is tracked by setting a bit in the change map. The default value is 64k (64KB). A smaller value requires more disk space for the change maps, but the finer granularity provides faster resynchronization.

You can also specify `vxassist`-style storage attributes to define the disks that can and/or cannot be used for the plexes of the DCO volume. You can specify the disks, controller and enclosure to be included or excluded in the `alloc` attributes. An example of the `alloc` usage to allocate storage from disks excluding `disk01` and `disk02`:

```
alloc=!disk01,!disk02
```

Note: The `vxsnap prepare` command automatically enables persistent FastResync on the volume. Persistent FastResync is also set automatically on any snapshots that are generated from a volume on which this feature is enabled.

If the volume is a RAID-5 volume, it is converted to a layered volume that can be used with instant snapshots and persistent FastResync.

By default, a new-style DCO volume contains 32 per-volume maps. If you require more maps than this, you can use the `vxsnap addmap` command to add more maps. See the `vxsnap(1M)` manual page for details of this command.

- 3 If you are going to create a snapshot volume by breaking off existing plexes, use the following command to add one or more snapshot mirrors to the volume:

```
# vxsnap [-b] [-g diskgroup] addmir volume [nmirror=N] \  
  [alloc=storage_attribute[,...]]
```

By default, one snapshot plex is added unless you specify a number using the `nmirror` attribute. For a backup, you should usually only require one plex. The mirrors remain in the `SNAPATT` state until they are fully synchronized. The `-b` option can be used to perform the synchronization in the background. Once synchronized, the mirrors are placed in the `SNAPDONE` state.

For example, the following command adds 2 mirrors to the volume, `vol1`, on disks `disk01` and `disk02`:

```
# vxsnap -g mydg addmir vol1 nmirror=2 alloc=disk01,disk02
```

Note: Do not perform this step if you create a full-sized snapshot volume using a suitably prepared empty volume, or if you create space-optimized snapshots that use a cache .

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

See [“Creating a shared cache object”](#) on page 151.

If the disks that contain volumes and their snapshots are to be moved into different disk groups, you must ensure that the disks that contain their DCO plexes can accompany them. You can use storage attributes to specify which disks to use for the DCO plexes. (If you do not want to use dirty region logging (DRL) with a volume, you can specify the same disks as those on which the volume is configured, assuming that space is available on the disks). For example, to add a DCO object and DCO volume with plexes on `disk05` and `disk06`, and a region size of 32KB, to the volume, `myvol`, use the following command:

```
# vxsnap -g mydg prepare myvol ndcomirs=2 regionsize=32k \  
  alloc=disk05,disk06
```

If required, you can use the `vxassist move` command to relocate DCO plexes to different disks. For example, the following command moves the plexes of the DCO volume for volume `vol1` from `disk03` and `disk04` to `disk07` and `disk08`:

```
# vxassist -g mydg move vol1_dc1 !disk03 !disk04 disk07 \  
  disk08
```

To view the details of the DCO object and DCO volume that are associated with a volume, use the `vxprint` command. The following is example `vxprint -vh` output for the volume named `zoo` (the `TUTIL0` and `PUTIL0` columns are omitted for clarity):

TTY	NAME	ASSOC	KSTATE	LENGTH	PLOFFS	STATE
...						
v	zoo	fsgen	ENABLED	1024	-	ACTIVE
pl	zoo-01	zoo	ENABLED	1024	-	ACTIVE
sd	disk01-01	zoo-01	ENABLED	1024	0	-
pl	zoo-02	zoo	ENABLED	1024	-	ACTIVE
sd	disk02-01	zoo-02	ENABLED	1024	0	-
dc	zoo_dco	zoo	-	-	-	-
v	zoo_dcl	gen	ENABLED	132	-	ACTIVE
pl	zoo_dcl-01	zoo_dcl	ENABLED	132	-	ACTIVE
sd	disk03-01	zoo_dcl-01	ENABLED	132	0	-
pl	zoo_dcl-02	zoo_dcl	ENABLED	132	-	ACTIVE
sd	disk04-01	zoo_dcl-02	ENABLED	132	0	-

In this output, the DCO object is shown as `zoo_dco`, and the DCO volume as `zoo_dcl` with 2 plexes, `zoo_dcl-01` and `zoo_dcl-02`.

See [“Considerations for placing DCO plexes”](#) on page 147.

For more information on DCO objects, see the `vxassist(1M)` and `vxsnap(1M)` manual pages.

Considerations for placing DCO plexes

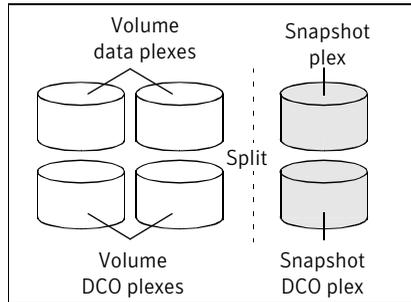
If you use the `vxassist` command or the Veritas Operations Manager (VOM) to create both a volume and its DCO, or the `vxsnap prepare` command to add a DCO to a volume, the DCO plexes are automatically placed on different disks from the data plexes of the parent volume. In previous releases, version 0 DCO plexes were placed on the same disks as the data plexes for convenience when performing disk group split and move operations. As the version 20 DCOs in VxVM 4.0 and later releases support dirty region logging (DRL) in addition to persistent FastResync, it is preferable for the DCO plexes to be separated from the data plexes. This improves the I/O performance of the volume, and provides resilience for the DRL logs.

If you use the `vxsnap prepare` command to set up a DCO, you must ensure that the disks that contain the plexes of the DCO volume accompany their parent volume during the move. Use the `vxprint` command on a volume to examine the configuration of its associated DCO volume.

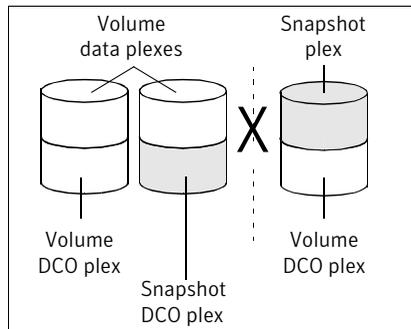
[Figure 12-1](#) illustrates some instances in which it is not possible to split a disk group because of the location of the DCO plexes. Relocate DCO plexes as needed.

See “[Preparing a volume for instant snapshot operations](#)” on page 143.

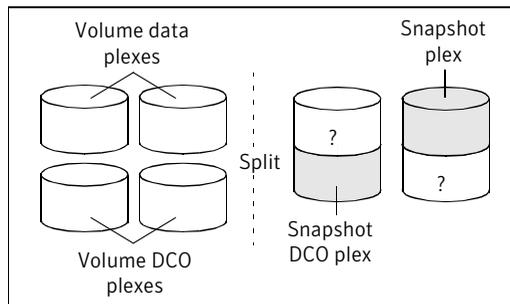
Figure 12-1 Examples of disk groups that can and cannot be split



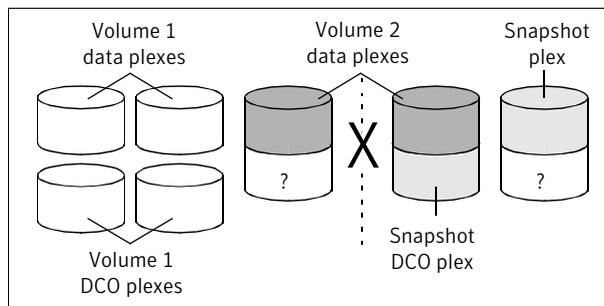
The disk group can be split as the DCO plexes are on dedicated disks, and can therefore accompany the disks that contain the volume data.



The disk group cannot be split as the DCO plexes cannot accompany their volumes. One solution is to relocate the DCO plexes. In this example, use an additional disk in the disk group as an intermediary to swap the misplaced DCO plexes. Alternatively, to improve DRL performance and resilience, allocate the DCO plexes to dedicated disks.



The disk group can be split as the DCO plexes can accompany their volumes. However, you may not wish the data in the portions of the disks marked “?” to be moved as well.



The disk group cannot be split as this would separate the disks that contain the data plexes of Volume 2. Possible solutions are to relocate the snapshot DCO plex to the disk containing the snapshot plex, or to another suitable disk that can be moved.

Creating a volume for use as a full-sized instant snapshot

If you want to create a full-sized instant snapshot for an original volume that does not contain any spare plexes, you can use an empty volume with the required degree of redundancy, and with the same size and same region size as the original volume.

To create an empty volume for use by a full-sized instant snapshot

- 1 Use the `vxprint` command on the original volume to find the required size for the snapshot volume.

```
# LEN=`vxprint [-g diskgroup] -F%len volume`
```

Note: The command shown in this and subsequent steps assumes that you are using a Bourne-type shell such as `sh`, `ksh` or `bash`. You may need to modify the command for other shells such as `csh` or `tcsh`. These steps are valid only for DCO version 20.

- 2 Use the `vxprint` command on the original volume to discover the name of its DCO:

```
# DCONAME=`vxprint [-g diskgroup] -F%dco_name volume`
```

- 3 Use the `vxprint` command on the DCO to discover its region size (in blocks):

```
# RSZ=`vxprint [-g diskgroup] -F%regionsz $DCONAME`
```

- 4 Use the `vxassist` command to create a volume, `snapvol`, of the required size and redundancy, together with a version 20 DCO volume with the correct region size:

```
# vxassist [-g diskgroup] make snapvol $LEN \
  [layout=mirror nmirror=number] logtype=dco dnl=no \
  dconversion=20 [ndcomirror=number] regionsz=$RSZ \
  init=active [storage_attributes]
```

It is recommended that you specify the same number of DCO mirrors (`ndcomirror`) as the number of mirrors in the volume (`nmirror`). The `init=active` attribute is used to make the volume available immediately. You can use storage attributes to specify which disks should be used for the volume.

As an alternative to creating the snapshot volume and its DCO volume in a single step, you can first create the volume, and then prepare it for instant snapshot operations as shown here:

```
# vxassist [-g diskgroup] make snapvol $LEN \
  [layout=mirror nmirror=number] init=active \
  [storage_attributes]
# vxsnap [-g diskgroup] prepare snapvol [ndcomirs=number] \
  regionsz=$RSZ [storage_attributes]
```

Creating a shared cache object

If you need to create several instant space-optimized snapshots for the volumes in a disk group, you may find it more convenient to create a single shared cache object in the disk group rather than a separate cache object for each snapshot.

To create a shared cache object

- 1 Decide on the following characteristics that you want to allocate to the cache volume that underlies the cache object:
 - The size of the cache volume should be sufficient to record changes to the parent volumes during the interval between snapshot refreshes. A suggested value is 10% of the total size of the parent volumes for a refresh interval of 24 hours.

- If redundancy is a desired characteristic of the cache volume, it should be mirrored. This increases the space that is required for the cache volume in proportion to the number of mirrors that it has.
 - If the cache volume is mirrored, space is required on at least as many disks as it has mirrors. These disks should not be shared with the disks used for the parent volumes. The disks should also be chosen to avoid impacting I/O performance for critical volumes, or hindering disk group split and join operations.
- 2 Having decided on its characteristics, use the `vxassist` command to create the volume that is to be used for the cache volume. The following example creates a mirrored cache volume, `cachevol`, with size 1GB in the disk group, `mydg`, on the disks `disk16` and `disk17`:

```
# vxassist -g mydg make cachevol 1g layout=mirror \  
  init=active disk16 disk17
```

The attribute `init=active` is specified to make the cache volume immediately available for use.

- 3 Use the `vxmake cache` command to create a cache object on top of the cache volume that you created in the previous step:

```
# vxmake [-g diskgroup] cache cache_object \  
cachevolname=volume [regionsize=size] [autogrow=on] \  
[highwatermark=hwmk] [autogrowby=agbvalue] \  
[maxautogrow=maxagbvalue]
```

If you specify the region size, it must be a power of 2, and be greater than or equal to 16KB (16k). If not specified, the region size of the cache is set to 64KB.

Note: All space-optimized snapshots that share the cache must have a region size that is equal to or an integer multiple of the region size set on the cache. Snapshot creation also fails if the original volume's region size is smaller than the cache's region size.

If the cache is not allowed to grow in size as required, specify `autogrow=off`. By default, the ability to automatically grow the cache is turned on.

In the following example, the cache object, `cobjmydg`, is created over the cache volume, `cachevol`, the region size of the cache is set to 32KB, and the `autogrow` feature is enabled:

```
# vxmake -g mydg cache cobjmydg cachevolname=cachevol \  
regionsize=32k autogrow=on
```

- 4 Having created the cache object, use the following command to enable it:

```
# vxcache [-g diskgroup] start cache_object
```

For example to start the cache object, `cobjmydg`:

```
# vxcache -g mydg start cobjmydg
```

Tuning the autogrow attributes

The `highwatermark`, `autogrowby` and `maxautogrow` attributes determine how the VxVM cache daemon (`vxcached`) maintains the cache if the `autogrow` feature has been enabled:

- When cache usage reaches the high watermark value, `highwatermark` (default value is 90 percent), and the new required cache size would not exceed the value of `maxautogrow` (default value is twice the size of the cache volume in

blocks), `vxcached` grows the size of the cache volume by the value of `autogrowby` (default value is 20% of the size of the cache volume in blocks).

- When cache usage reaches the high watermark value, and the new required cache size would exceed the value of `maxautogrow`, `vxcached` deletes the oldest snapshot in the cache. If there are several snapshots with the same age, the largest of these is deleted.

If the `autogrow` feature has been disabled:

- When cache usage reaches the high watermark value, `vxcached` deletes the oldest snapshot in the cache. If there are several snapshots with the same age, the largest of these is deleted. If there is only a single snapshot, this snapshot is detached and marked as invalid.

Note: The `vxcached` daemon does not remove snapshots that are currently open, and it does not remove the last or only snapshot in the cache.

If the cache space fills up, the snapshot is detached and marked as invalid. If this happens, the snapshot is unrecoverable and must be removed. Enabling the `autogrow` feature on the cache helps to avoid this situation from occurring.

However, for very small caches (of the order of a few megabytes), it is possible for the cache to become exhausted before the system has time to respond and grow the cache. In such cases, use the `vxcache` command to increase the size of the cache, or to reduce the value of `highwatermark`.

If necessary, you can use the `vxcache set` command to change other `autogrow` attribute values for a cache. For example, you can use the `maxautogrow` attribute to limit the maximum size to which a cache can grow. To estimate this size, consider how much the contents of each source volume are likely to change between snapshot refreshes, and allow some additional space for contingency.

Warning: Ensure that the cache is sufficiently large, and that the `autogrow` attributes are configured correctly for your needs.

See the `vxcache(1M)` manual page and the “Administering Volume Snapshots” chapter in the *Veritas Volume Manager Administrator’s Guide* for more information including how to grow, shrink and remove a storage cache.

Online database backup

This chapter includes the following topics:

- [About online database backup](#)
- [Making a backup of an online database on the same host](#)
- [Making an off-host backup of an online database](#)

About online database backup

Online backup of a database can be implemented by configuring either the primary host or a dedicated separate host to perform the backup operation on snapshot mirrors of the primary host's database.

Two backup methods are described in the following sections:

- See [“Making a backup of an online database on the same host”](#) on page 156.
- See [“Making an off-host backup of an online database”](#) on page 160.

Note: All commands require superuser (`root`) or equivalent privileges, except where it is explicitly stated that a command must be run by the database administrator.

The following sections include sample scripts:

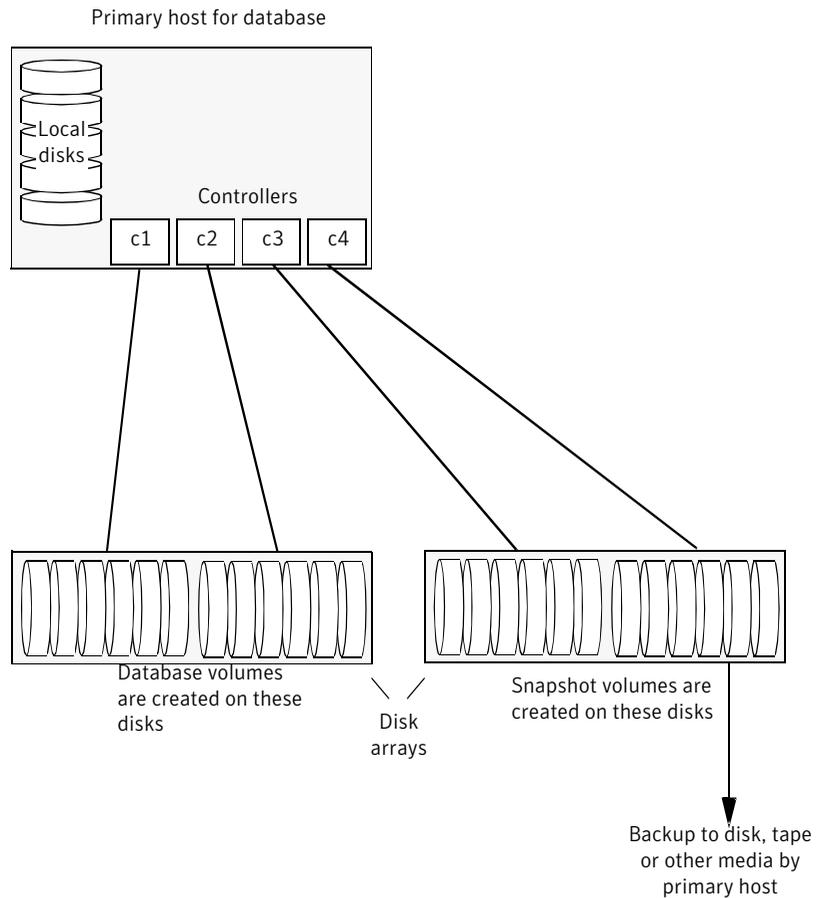
- See [“Script to initiate online off-host backup of an Oracle database”](#) on page 509.
- See [“Script to put an Oracle database into hot backup mode”](#) on page 511.
- See [“Script to quiesce a Sybase ASE database”](#) on page 512.
- See [“Script to suspend I/O for a DB2 database”](#) on page 512.
- See [“Script to end Oracle database hot backup mode”](#) on page 513.

- See [“Script to release a Sybase ASE database from quiesce mode”](#) on page 513.
- See [“Script to resume I/O for a DB2 database”](#) on page 514.
- See [“Script to perform off-host backup”](#) on page 514.

Making a backup of an online database on the same host

[Figure 13-1](#) shows an example with two primary database volumes to be backed up, `dbase_vol` and `dbase_logs`, which are configured on disks attached to controllers `c1` and `c2`, and the snapshots to be created on disks attached to controllers `c3` and `c4`.

Figure 13-1 Example system configuration for database backup on the primary host



Note: It is assumed that you have already prepared the volumes containing the file systems for the datafiles to be backed up as described in the example.

See [“About setting up volumes for instant snapshots”](#) on page 141.

For an Oracle database, it is not necessary to create snapshots of the volumes containing the file systems for the redo log volumes or archived logs.

To make a backup of an online database on the same host

- 1 Use the following commands to add one or more snapshot plexes to the volume, and to make a full-sized break-off snapshot, *snapvol*, of the tablespace volume by breaking off these plexes:

```
# vxsnap -g volumedg addmir volume [nmirror=N] \
  [alloc=storage_attributes]
# vxsnap -g volumedg make \
  source=volume/newvol=snapvol[/nmirror=N] \
  [alloc=storage_attributes]
```

By default, one snapshot plex is added unless you specify a number using the `nmirror` attribute. For a backup, you should usually only require one plex. You can specify storage attributes (such as a list of disks) to determine where the plexes are created.

If the volume layout does not support plex break-off, prepare an empty volume for the snapshot.

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

Then use the following command to create the snapshot:

```
# vxsnap -g volumedg make source=volume/snapvol=snapvol
```

If a database spans more than one volume, specify all the volumes and their snapshot volumes as separate tuples on the same line, for example:

```
# vxsnap -g dbasedg make source=vol1/snapvol=svol1 \
  source=vol2/newvol=svol2 source=vol3/snapvol=svol3
```

If you want to save disk space, you can use the following command to create a space-optimized snapshot instead:

```
# vxsnap -g volumedg make \
  source=volume/newvol=snapvol/cache=cacheobject
```

The argument `cacheobject` is the name of a pre-existing cache that you have created in the disk group for use with space-optimized snapshots.

See [“Creating a shared cache object”](#) on page 151.

If several space-optimized snapshots are to be created at the same time, these can all specify the same cache object as shown in this example:

```
# vxsnap -g dbasedg make \
  source=vol1/newvol=svol1/cache=dbaseco \
  source=vol2/newvol=svol2/cache=dbaseco \
  source=vol3/newvol=svol3/cache=dbaseco
```

Note: This step sets up the snapshot volumes ready for the backup cycle, and starts tracking changes to the original volumes.

When you are ready to make a backup, proceed to step 2.

- 2 If the volumes to be backed up contain database tables in file systems, suspend updates to the volumes:
 - DB2 provides the `write suspend` command to temporarily suspend I/O activity for a database. As the DB2 database administrator, use a script such as that shown in the example.
 See [“Script to suspend I/O for a DB2 database”](#) on page 512.
 Note that to allow recovery from any backups taken from snapshots, the database must be in LOGRETAIN RECOVERY mode.
 - Oracle supports online backup by temporarily suspending updates to the datafiles of the tablespaces, provided that the database is running in archive mode and the tablespaces are online. As the Oracle database administrator, put each tablespace into hot backup mode using a script such as that shown in the example.
 See [“Script to put an Oracle database into hot backup mode”](#) on page 511.
 - Sybase ASE from version 12.0 onward provides the Quiesce feature to allow temporary suspension of writes to a database. As the Sybase database administrator, put the database in quiesce mode by using a script such as that shown in the example.
 See [“Script to quiesce a Sybase ASE database”](#) on page 512.
- 3 Refresh the contents of the snapshot volumes from the original volume using the following command:

```
# vxsnap -g volumedg refresh snapvol source=vol \
  [snapvol2 source=vol2] ...
```

For example, to refresh the snapshots `svol1`, `svol2` and `svol3`:

```
# vxsnap -g dbasedg refresh svol1 source=vol1 \
  svol2 source=vol2 svol3 source=vol3
```

- 4 If you have temporarily suspended updates to volumes in 2:
 - As the DB2 database administrator, use a script such as that shown in the example.
 See [“Script to resume I/O for a DB2 database”](#) on page 514.
 - As the Oracle database administrator, release all the tablespaces from hot backup mode using a script such as that shown in the example.
 See [“Script to end Oracle database hot backup mode”](#) on page 513.
 - As the Sybase database administrator, release the database from quiesce mode using a script such as that shown in the example.

See “[Script to release a Sybase ASE database from quiesce mode](#)” on page 513.

- 5 Back up the snapshot volume. If you need to remount the file system in the volume to back it up, first run `fscck` on the volume. The following are sample commands for checking and mounting a file system:

```
# fscck -V vxfs /dev/vx/rdsk/snapvoldg/snapvol
# mount -V vxfs /dev/vx/dsk/snapvoldg/snapvol
    mount_point
```

Back up the file system at this point using a command such as `bpbbackup` in Symantec NetBackup. After the backup is complete, use the following command to unmount the file system.

```
# umount mount_point
```

Repeat steps 2 through 5 each time that you need to back up the volume.

In some instances, such as recovering the contents of a corrupted volume, it may be useful to resynchronize a volume from its snapshot volume (which is used as a hot standby):

```
# vxsnap -g diskgroup restore volume source=snapvol \
    destroy=yes|no
```

The `destroy` attribute specifies whether the plexes of the snapshot volume are to be reattached to the original volume. For example, to resynchronize the volume `dbase_vol` from its snapshot volume `snap2_dbase_vol` without removing the snapshot volume:

```
# vxsnap -g dbasedg restore dbase_vol \
    source=snap2_dbase_vol destroy=no
```

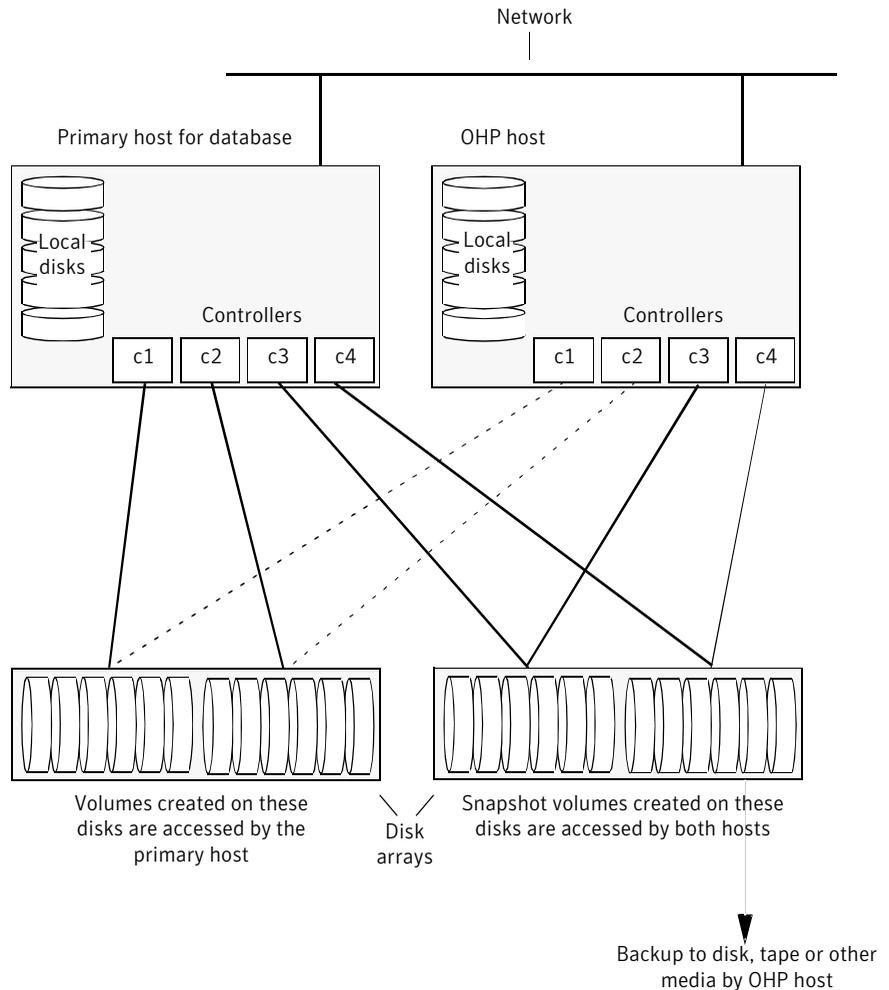
Note: You must shut down the database and unmount the file system that is configured on the original volume before attempting to resynchronize its contents from a snapshot.

Making an off-host backup of an online database

Figure 13-2 shows an example of two primary database volumes to be backed up, `dbase_vol` and `dbase_logs`, which are configured on disks attached to controllers `c1` and `c2`, and the snapshots to be created on disks attached to controllers `c3` and `c4`.

There is no requirement for the OHP host to have access to the disks that contain the primary database volumes.

Figure 13-2 Example system configuration for off-host database backup



Note: It is assumed that you have already prepared the volumes containing the file systems for the datafiles to be backed up as described in the example.

See [“About setting up volumes for instant snapshots”](#) on page 141.

For an Oracle database, it is not necessary to create snapshots of the volumes containing the file systems for the redo log volumes or archived logs.

If the database is configured on volumes in a cluster-shareable disk group, it is assumed that the primary host for the database is the master node for the cluster. However, if the primary host is not also the master node, most Volume Manager operations on shared disk groups are best performed on the master node.

The procedure in this section is designed to minimize copy-on-write operations that can impact system performance. You can also implement this procedure on a single host by omitting steps 5 through [To make an off-host backup of an online database](#) and 10 through 13 that split, deport, reimport and rejoin the snapshot disk group.

To make an off-host backup of an online database

- 1 On the primary host, add one or more snapshot plexes to the volume using this command:

```
# vxsnap -g volumedg admir volume [nmirror=N] \  
  [alloc=storage_attributes]
```

By default, one snapshot plex is added unless you specify a number using the `nmirror` attribute. For a backup, you should usually only require one plex. You can specify storage attributes (such as a list of disks) to determine where the plexes are created.

- 2 Suspend updates to the volumes:

- DB2 provides the `write suspend` command to temporarily suspend I/O activity for a database. As the DB2 database administrator, use a script such as that shown in the example.
See [“Script to suspend I/O for a DB2 database”](#) on page 512.
Note that to allow recovery from any backups taken from snapshots, the database must be in LOGRETAIN RECOVERY mode.
- Oracle supports online backup by temporarily suspending updates to the datafiles of the tablespaces, provided that the database is running in archive mode and the tablespaces are online. As the Oracle database administrator, put each tablespace into hot backup mode using a script such as that shown in the example.
See [“Script to put an Oracle database into hot backup mode”](#) on page 511.
- Sybase ASE from version 12.0 onward provides the Quiesce feature to allow temporary suspension of writes to a database. As the Sybase database administrator, put the database in quiesce mode by using a script such as that shown in the example.
See [“Script to quiesce a Sybase ASE database”](#) on page 512.

- 3 Use the following command to make a full-sized snapshot, *snapvol*, of the tablespace volume by breaking off the plexes that you added in step 1 from the original volume:

```
# vxsnap -g volumedg make \  

   source=volume/newvol=snapvol/nmirror=N \  

   [alloc=storage_attributes]
```

The `nmirror` attribute specifies the number of mirrors, *N*, in the snapshot volume.

If a database spans more than one volume, specify all the volumes and their snapshot volumes as separate tuples on the same line, for example:

```
# vxsnap -g dbasedg make source=vol1/newvol=svol1 \  

   source=vol2/newvol=svol2 source=vol3/newvol=svol3 \  

   alloc=ctlr:c3,ctlr:c4
```

This step sets up the snapshot volumes ready for the backup cycle, and starts tracking changes to the original volumes.

- 4 Release all the tablespaces or databases from suspend, hot backup or quiesce mode:
 - As the DB2 database administrator, use a script such as that shown in the example.
 See [“Script to resume I/O for a DB2 database”](#) on page 514.
 - As the Oracle database administrator, release all the tablespaces from hot backup mode using a script such as that shown in the example.
 See [“Script to end Oracle database hot backup mode”](#) on page 513.
 - As the Sybase database administrator, release the database from quiesce mode using a script such as that shown in the example.
 See [“Script to release a Sybase ASE database from quiesce mode”](#) on page 513.
- 5 On the primary host, use the following command to split the disks containing the snapshot volumes into a separate disk group, *snapvoldg*, from the original disk group, *volumedg*:

```
# vxdg split volumedg
   snapvoldg
   snapvol ...
```

- 6 On the primary host, deport the snapshot volume's disk group using the following command:

```
# vxdg deport snapvoldg
```

- 7 On the OHP host where the backup is to be performed, use the following command to import the snapshot volume's disk group:

```
# vxdg import snapvoldg
```

- 8 VxVM will recover the volumes automatically after the disk group import unless it is set to not recover automatically. Check if the snapshot volume is initially disabled and not recovered following the split.

If a volume is in the DISABLED state, use the following command on the OHP host to recover and restart the snapshot volume:

```
# vxrecover -g snapvoldg -m snapvol ...
```

- 9 On the OHP host, back up the snapshot volumes. If you need to remount the file system in the volume to back it up, first run `fsck` on the volumes. The following are sample commands for checking and mounting a file system:

```
# fsck -V vxfs /dev/vx/rdisk/snapvoldg/snapvol  
# mount -V vxfs /dev/vx/dsk/snapvoldg/snapvol  
    mount_point
```

Back up the file system using a command such as `bpbbackup` in Symantec NetBackup. After the backup is complete, use the following command to unmount the file system.

```
# umount mount_point
```

- 10 On the OHP host, use the following command to deport the snapshot volume's disk group:

```
# vxdg deport snapvoldg
```

- 11 On the primary host, re-import the snapshot volume's disk group using the following command:

```
# vxdg [-s] import snapvoldg
```

Note: Specify the `-s` option if you are reimporting the disk group to be rejoined with a shared disk group in a cluster.

- 12** On the primary host, use the following command to rejoin the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join snapvoldg
      volumedg
```

- 13** VxVM will recover the volumes automatically after the join unless it is set not to recover automatically. Check if the snapshot volumes are initially disabled and not recovered following the join.

If a volume is in the DISABLED state, use the following command on the primary host to recover and restart the snapshot volume:

```
# vxrecover -g volumedg -m snapvol
```

- 14** On the primary host, reattach the snapshot volumes to their original volume using the following command:

```
# vxsnap -g volumedg reattach snapvol source=vol \
      [snapvol2 source=vol2]...
```

For example, to reattach the snapshot volumes `svol1`, `svol2` and `svol3`:

```
# vxsnap -g dbasedg reattach svol1 source=vol1 \
      svol2 source=vol2 svol3 source=vol3
```

While the reattached plexes are being resynchronized from the data in the parent volume, they remain in the `SNAPTMP` state. After resynchronization is complete, the plexes are placed in the `SNAPDONE` state. You can use the `vxsnap print` command to check on the progress of synchronization.

Repeat steps 2 through 14 each time that you need to back up the volume.

This guide provides an example of a script that uses this method.

See [“Script to initiate online off-host backup of an Oracle database”](#) on page 509.

In some instances, such as recovering the contents of a corrupted volume, it may be useful to resynchronize a volume from its snapshot volume (which is used as a hot standby):

```
# vxsnap -g diskgroup restore volume source=snapvol \
      destroy=yes|no
```

The `destroy` attribute specifies whether the plexes of the snapshot volume are to be reattached to the original volume. For example, to resynchronize the volume

dbase_vol from its snapshot volume snap2_dbase_vol without removing the snapshot volume:

```
# vxsnap -g dbasedg restore dbase_vol \  
    source=snap2_dbase_vol destroy=no
```

Note: You must shut down the database and unmount the file system that is configured on the original volume before attempting to resynchronize its contents from a snapshot.

Off-host cluster file system backup

This chapter includes the following topics:

- [About off-host cluster file system backup](#)
- [Mounting a file system for shared access](#)
- [Using off-host processing to back up cluster file systems](#)

About off-host cluster file system backup

Veritas Cluster File System (CFS) allows cluster nodes to share access to the same file system. CFS is especially useful for sharing read-intensive data between cluster nodes.

Off-host backup of cluster file systems may be implemented by taking a snapshot of the volume containing the file system and performing the backup operation on a separate host.

[Figure 14-1](#) shows an example where the primary volume that contains the file system to be backed up is configured on disks attached to controllers `c1` and `c2`, and the snapshots are to be created on disks attached to controllers `c3` and `c4`.

You can mount a VxFS file system for shared access by the nodes of a cluster.

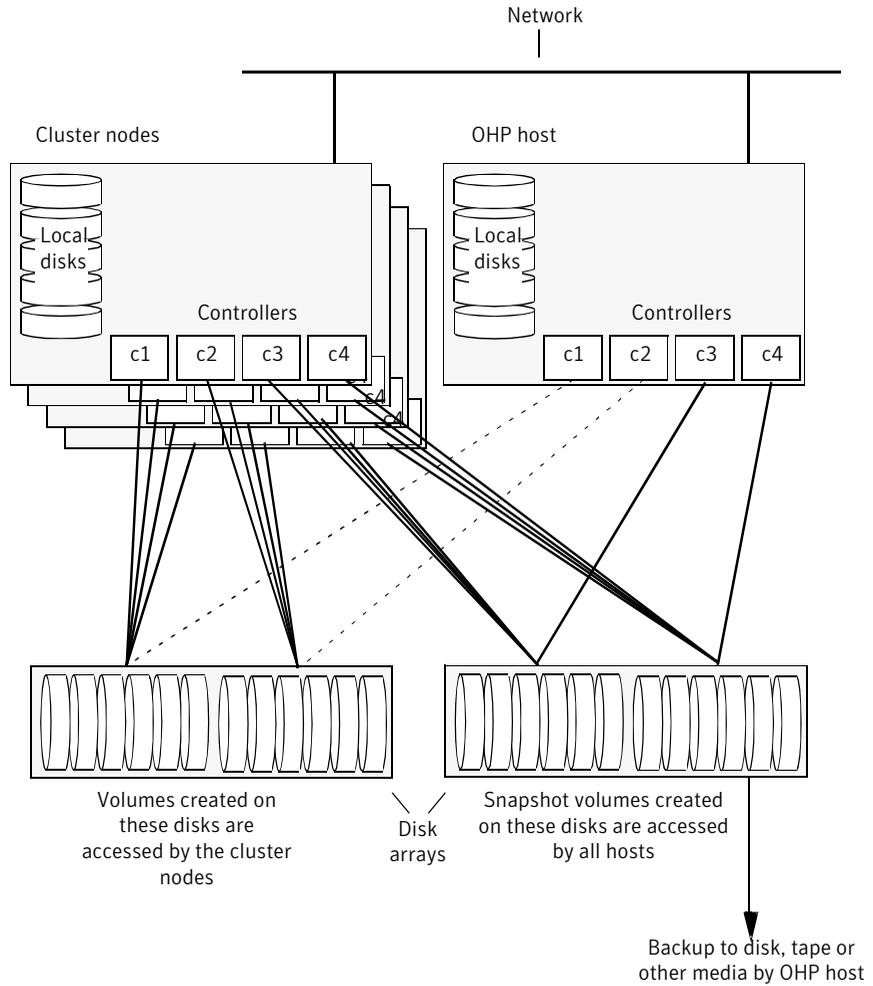
See [“Mounting a file system for shared access”](#) on page 168.

You can perform off-host backups of cluster-shared file systems.

See [“Using off-host processing to back up cluster file systems”](#) on page 169.

Note: All commands require superuser (`root`) or equivalent privileges.

Figure 14-1 System configuration for off-host file system backup scenarios



Mounting a file system for shared access

To mount a VxFS file system for shared access, use the following command on each cluster node where required:

```
# mount -V vxfs -o cluster /dev/vx/dsk/diskgroup/volume  
mount_point
```

For example, to mount the volume `cfs_vol` in the disk group `exampledg` for shared access on the mount point, `/mnt_pnt`:

```
# mount -V vxfs -o cluster /dev/vx/dsk/exampledg/cfs_vol /mnt_pnt
```

Using off-host processing to back up cluster file systems

Before using this procedure, you must prepare the volumes containing the file systems that are to be backed up.

See [“About setting up volumes for instant snapshots”](#) on page 141.

Warning: This procedure assumes that you have already prepared the volumes containing the file systems that are to be backed up.

While you can run the commands in the following steps from any node, Symantec recommends running them from the master node.

To back up a snapshot of a mounted file system which has shared access

- 1 On the master node, use the following command to make a full-sized snapshot, *snapvol*, of the volume containing the file system by breaking off plexes from the original volume:

```
# vxsnap -g volumedg make \  
  source=volume/newvol=snapvol/nmirror=N
```

The `nmirror` attribute specifies the number of mirrors, *N*, in the snapshot volume.

For example, to take a snapshot of the volume *cfs_vol* in the shared disk group *exampledg*:

```
# vxsnap -g exampledg make source=cfs_vol/newvol=scfs_vol \  
/nmirror=1
```

If the volume does not have any available plexes, or its layout does not support plex break-off, prepare an empty volume for the snapshot.

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

Then use the following command to create the snapshot:

```
# vxsnap -g volumedg make source=volume/snapvol=snapvol
```

Note: This step actually takes the snapshot and sets up the snapshot volumes ready for the backup cycle, and starts tracking changes to the original volumes.

When you are ready to make a backup, proceed to step 2.

- 2 On any node, refresh the contents of the snapshot volumes from the original volume using the following command:

```
# vxsnap -g volumedg refresh snapvol source=vol \  
[snapvol2 source=vol2]... syncing=yes
```

The `syncing=yes` attribute starts a synchronization of the snapshot in the background.

For example, to refresh the snapshot `scfs_vol`:

```
# vxsnap -g exampledg refresh scfs_vol source=cfs_vol \  
syncing=yes
```

This command can be run every time you want to back up the data. The `vxsnap refresh` command will resync only those regions which have been modified since the last refresh.

- 3 On any node of the cluster, use the following command to wait for the contents of the snapshot to be fully synchronous with the contents of the original volume:

```
# vxsnap -g volumedg syncwait snapvol
```

For example, to wait for synchronization to finish for the snapshots `scfs_vol`:

```
# vxsnap -g exampledg syncwait scfs_vol
```

Note: You cannot move a snapshot volume into a different disk group until synchronization of its contents is complete. You can use the `vxsnap print` command to check on the progress of synchronization.

- 4 On the master node, use the following command to split the snapshot volume into a separate disk group, `snapvoldg`, from the original disk group, `volumedg`:

```
# vxdg split volumedg  
snapvoldg  
snapvol
```

For example, to place the snapshot of the volume `cfs_vol` into the shared disk group `splitdg`:

```
# vxdg split exampledg splitdg scfs_vol
```

- 5 On the master node, deport the snapshot volume's disk group using the following command:

```
# vxdg deport snapvoldg
```

For example, to deport the disk group *splitdg*:

```
# vxdg deport splitdg
```

- 6 On the OHP host where the backup is to be performed, use the following command to import the snapshot volume's disk group:

```
# vxdg import snapvoldg
```

For example, to import the disk group *splitdg*:

```
# vxdg import splitdg
```

- 7 VxVM will recover the volumes automatically after the disk group import unless it is set not to recover automatically. Check if the snapshot volume is initially disabled and not recovered following the split.

If a volume is in the DISABLED state, use the following command on the OHP host to recover and restart the snapshot volume:

```
# vxrecover -g snapvoldg -m snapvol
```

For example, to start the volume *snapvol*:

```
# vxrecover -g splitdg -m snapvol
```

- 8 On the OHP host, use the following commands to check and locally mount the snapshot volume:

```
# fsck -V vxfs /dev/vx/rdisk/diskgroup/volume
```

```
# mount -V vxfs /dev/vx/dsk/diskgroup/volume  
mount_point
```

For example, to check and mount the volume *scfs_vol* in the disk group *splitdg* for shared access on the mount point, */bak/mnt_pnt*:

```
# fsck -V vxfs /dev/vx/rdisk/splitdg/scfs_vol
```

```
# mount -V vxfs /dev/vx/dsk/splitdg/scfs_vol /bak/mnt_pnt
```

- 9** Back up the file system at this point using a command such as `bpbackup` in Symantec NetBackup. After the backup is complete, use the following command to unmount the file system.

```
# umount mount_point
```

- 10** On the OHP host, use the following command to deport the snapshot volume's disk group:

```
# vxdg deport snapvoldg
```

For example, to deport *splitdg*:

```
# vxdg deport splitdg
```

- 11** On the master node, re-import the snapshot volume's disk group as a shared disk group using the following command:

```
# vxdg -s import snapvoldg
```

For example, to import *splitdg*:

```
# vxdg -s import splitdg
```

- 12** On the master node, use the following command to rejoin the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join snapvoldg volumedg
```

For example, to join disk group *splitdg* with *exampledg*:

```
# vxdg join splitdg exampledg
```

- 13** VxVM will recover the volumes automatically after the join unless it is set not to recover automatically. Check if the snapshot volumes are initially disabled and not recovered following the join.

If a volume is in the DISABLED state, use the following command on the primary host to recover and restart the snapshot volume:

```
# vxrecover -g volumedg -m snapvol
```

- 14** When the backup is complete, use the following command to unmount the snapshot volume, and make it ready for its contents to be refreshed from the primary volume:

```
# umount mount_point
```

When synchronization is complete, the snapshot is ready to be re-used for backup.

Warning: Before attempting to unmount the snapshot, shut down all applications that access a file system in the snapshot volume, and also unmount any such file system.

Repeat steps 2 through 14 each time that you need to back up the volume.

In some instances, such as recovering the contents of a corrupted volume, it may be useful to resynchronize a volume from its snapshot volume (which is used as a hot standby):

```
# vxsnap -g diskgroup restore volume source=snapvol  
destroy=yes|no
```

The `destroy` attribute specifies whether the plexes of the snapshot volume are to be reattached to the original volume. For example, to resynchronize the volume `cfs_vol` from its snapshot volume `scfs_vol`:

```
# vxsnap -g exampledg restore cfs_vol source=scfs_vol destroy=no
```

Note: You must unmount the file system that is configured on the original volume before attempting to resynchronize its contents from a snapshot.

Reattaching snapshot plexes

Some or all plexes of an instant snapshot may be reattached to the specified original volume, or to a source volume in the snapshot hierarchy above the snapshot volume.

Note: This operation is not supported for space-optimized instant snapshots.

By default, all the plexes are reattached, which results in the removal of the snapshot. If required, the number of plexes to be reattached may be specified as the value assigned to the `nmirror` attribute.

Note: The snapshot being reattached must not be open to any application. For example, any file system configured on the snapshot volume must first be unmounted.

To reattach a snapshot

- ◆ Use the following command to reattach an instant snapshot to the specified original volume, or to a source volume in the snapshot hierarchy above the snapshot volume:

```
# vxsnap [-g diskgroup] reattach snapvol source=vol \  
[nmirror=number]
```

For example the following command reattaches 1 plex from the snapshot volume, *snapmyvol*, to the volume, *myvol*:

```
# vxsnap -g mydg reattach snapmyvol source=myvol nmirror=1
```

While the reattached plexes are being resynchronized from the data in the parent volume, they remain in the `SNAPTMP` state. After resynchronization is complete, the plexes are placed in the `SNAPDONE` state.

The `vxsnap refresh` and `vxsnap reattach` commands have slightly different behaviors.

The `vxsnap reattach` command reattaches a snapshot volume to its source volume and begins copying the volume data to the snapshot volume.

The `vxsnap refresh` command updates the snapshot volumes contents view. The updated snapshot is available immediately with the new contents while synchronization occurs in the background.

Decision support

This chapter includes the following topics:

- [About decision support](#)
- [Creating a replica database on the same host](#)
- [Creating an off-host replica database](#)

About decision support

You can use snapshots of a primary database to create its replica at a given moment in time. You can then implement decision support analysis and report generation operations that take their data from the database copy rather than from the primary database. The FastResync functionality of Veritas Volume Manager (VxVM) allows you to quickly refresh the database copy with up-to-date information from the primary database. Reducing the time taken to update decision support data also lets you generate analysis reports more frequently.

Two methods are described for setting up a replica database for decision support:

- See [“Creating a replica database on the same host”](#) on page 178.
- See [“Creating an off-host replica database”](#) on page 184.

Note: All commands require superuser (`root`) or equivalent privileges, except where it is explicitly stated that a command must be run by the database administrator.

The following sections include sample scripts:

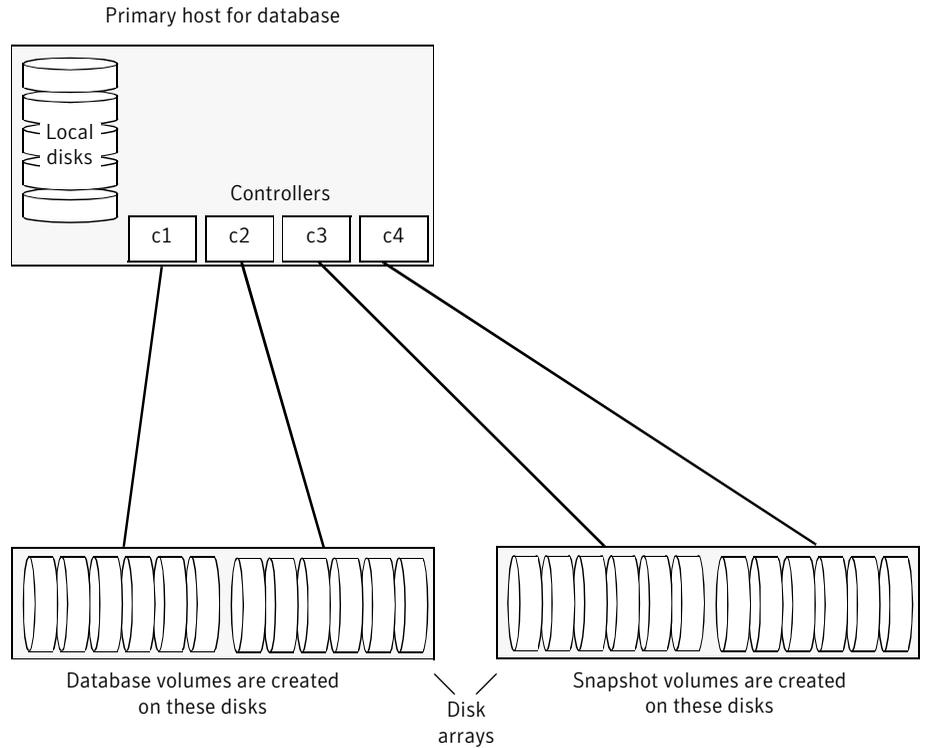
- See [“Script to put an Oracle database into hot backup mode”](#) on page 511.
- See [“Script to quiesce a Sybase ASE database”](#) on page 512.

- See [“Script to suspend I/O for a DB2 database”](#) on page 512.
- See [“Script to end Oracle database hot backup mode”](#) on page 513.
- See [“Script to release a Sybase ASE database from quiesce mode”](#) on page 513.
- See [“Script to resume I/O for a DB2 database”](#) on page 514.
- See [“Script to create an off-host replica Oracle database”](#) on page 515.
- See [“Script to complete, recover and start a replica Oracle database”](#) on page 517.
- See [“Script to start a replica Sybase ASE database”](#) on page 519.

Creating a replica database on the same host

[Figure 15-1](#) shows an example where the primary database volumes to be backed up, `dbase_vol` and `dbase_logs`, are configured on disks attached to controllers `c1` and `c2`, and the snapshots are to be created on disks attached to controllers `c3` and `c4`.

Figure 15-1 Example system configuration for decision support on the primary host



Note: It is assumed that you have already prepared the database volumes to be replicated as described in the example.

See [“About setting up volumes for instant snapshots”](#) on page 141.

To set up a replica database to be used for decision support on the primary host

- 1** Prepare, if you have not already done so, the host to use the snapshot volume that contains the copy of the database tables. Set up any new database logs and configuration files that are required to initialize the database.

- 2 Use the following command to make a full-sized snapshot, *snapvol*, of the tablespace volume by breaking off plexes from the original volume:

```
# vxsnap -g volumedg make \  
    source=volume/newvol=snapvol/nmirror=N
```

The `nmirror` attribute specifies the number of mirrors, *N*, in the snapshot volume.

If the volume does not have any available plexes, or its layout does not support plex break-off, prepare an empty volume for the snapshot.

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

Then use the following command to create the snapshot:

```
# vxsnap -g volumedg make source=volume/snapvol=snapvol
```

If a database spans more than one volume, specify all the volumes and their snapshot volumes as separate tuples on the same line, for example:

```
# vxsnap -g dbasedg make \  
source=vol1/snapvol=svol1/nmirror=2 \  
source=vol2/snapvol=svol2/nmirror=2 \  
source=vol3/snapvol=svol3/nmirror=2
```

If you want to save disk space, you can use the following command to create a space-optimized snapshot instead:

```
# vxsnap -g volumedg make \  
    source=volume/newvol=snapvol/cache=cacheobject
```

The argument `cacheobject` is the name of a pre-existing cache that you have created in the disk group for use with space-optimized snapshots.

See [“Creating a shared cache object”](#) on page 151.

If several space-optimized snapshots are to be created at the same time, these can all specify the same cache object as shown in this example:

```
# vxsnap -g dbasedg make \  
    source=vol1/newvol=svol1/cache=dbaseco \  
    source=vol2/newvol=svol2/cache=dbaseco \  
    source=vol3/newvol=svol3/cache=dbaseco
```

See the section “Creating a Share Cache Object” in the “Administering Volume Snapshots” chapter of the *Veritas Volume Manager Administrator’s Guide* for more information.

Note: This step sets up the snapshot volumes, and starts tracking changes to the original volumes.

When you are ready to create the replica database, proceed to step 3.

- 3 If the volumes to be backed up contain database tables in file systems, suspend updates to the volumes:
 - DB2 provides the `write suspend` command to temporarily suspend I/O activity for a database. As the DB2 database administrator, use a script such as the example script.
See [“Script to suspend I/O for a DB2 database”](#) on page 512.
Note that to allow recovery from any backups taken from snapshots, the database must be in LOGRETAIN RECOVERY mode.
 - Oracle supports online backup by temporarily suspending updates to the datafiles of the tablespaces, provided that the database is running in archive mode and the tablespaces are online. As the Oracle database administrator, put each tablespace into hot backup mode using a script such as the example script.
See [“Script to put an Oracle database into hot backup mode”](#) on page 511.
 - Sybase ASE from version 12.0 onward provides the Quiesce feature to allow temporary suspension of writes to a database. As the Sybase database administrator, put the database in quiesce mode by using a script such as the example script.
See [“Script to quiesce a Sybase ASE database”](#) on page 512.
If you are using Sybase ASE 12.5, you can specify the `for external dump` clause to the `quiesce` command. This warm standby method allows you to update a replica database using transaction logs dumped from the primary database.
See [“Updating a warm standby Sybase ASE 12.5 database”](#) on page 192.
- 4 Refresh the contents of the snapshot volumes from the original volume using the following command:

```
# vxsnap -g volumedg refresh snapvol source=vol \  
[snapvol2 source=vol2]...
```

For example, to refresh the snapshots `svol1`, `svol2` and `svol3`:

```
# vxsnap -g dbasedg refresh svol1 source=vol1 \  
svol2 source=vol2 svol3 source=vol3
```

- 5 If you temporarily suspended updates to volumes in step 3, perform the following steps.

Release all the tablespaces or databases from suspend, hot backup, or quiesce mode:

- As the DB2 database administrator, use a script such as the example script. See “[Script to resume I/O for a DB2 database](#)” on page 514.
- As the Oracle database administrator, release all the tablespaces from hot backup mode using a script such as the example script. See “[Script to end Oracle database hot backup mode](#)” on page 513.
- As the Sybase database administrator, release the database from quiesce mode using a script such as the example script. See “[Script to release a Sybase ASE database from quiesce mode](#)” on page 513.

If you are using Sybase ASE 12.5, you can specify the `for external dump` clause to the `quiesce` command. This warm standby method allows you to update a replica database using transaction logs dumped from the primary database.

See “[Updating a warm standby Sybase ASE 12.5 database](#)” on page 192.

- 6 For each snapshot volume containing tablespaces, check the file system that it contains, and mount the volume using the following commands:

```
# fsck -V vxfs /dev/vx/rdisk/diskgroup/snapvol
# mount -V vxfs /dev/vx/dsk/diskgroup/snapvol
    mount_point
```

For example, to check the file system in the snapshot volume `snap1_dbase_vol`, and mount it on `/rep_dbase_vol`:

```
# fsck -V vxfs /dev/vx/rdisk/dbasedg/snap1_dbase_vol
# mount -V vxfs /dev/vx/dsk/dbasedg/snap1_dbase_vol \
    /rep_dbase_vol
```

- 7 Copy any required log files from the primary database to the replica database.
- For an Oracle database, copy the archived log files that were generated while the database was in hot backup mode to the new database’s archived log directory (for example, `/rep_archlog`).
 - For a Sybase ASE database, if you specified the `for external dump` clause when you quiesced the database, use the following `isql` command as the database administrator to dump the transaction log for the database:

```
dump transaction to dump_device with standby_access
```

Then copy the dumped transaction log to the appropriate replica database directory.

- 8 As the database administrator, start the new database:
 - For an Oracle database, use a script such as the example script. See “[Script to complete, recover and start a replica Oracle database](#)” on page 517.
 - For a Sybase ASE database, use a script such as the example script. See “[Script to start a replica Sybase ASE database](#)” on page 519. If you are using the warm standby method, specify the `-q` option to the `dataserver` command. Use the following `isql` commands to load the dump of the transaction log and put the database online:

```
load transaction from dump_device with standby_access  
online database database_name for standby_access
```

If you are not using the warm standby method, use the following `isql` command to recover the database, roll back any uncommitted transactions to the time that the quiesce command was issued, and put the database online:

```
online database database_name
```

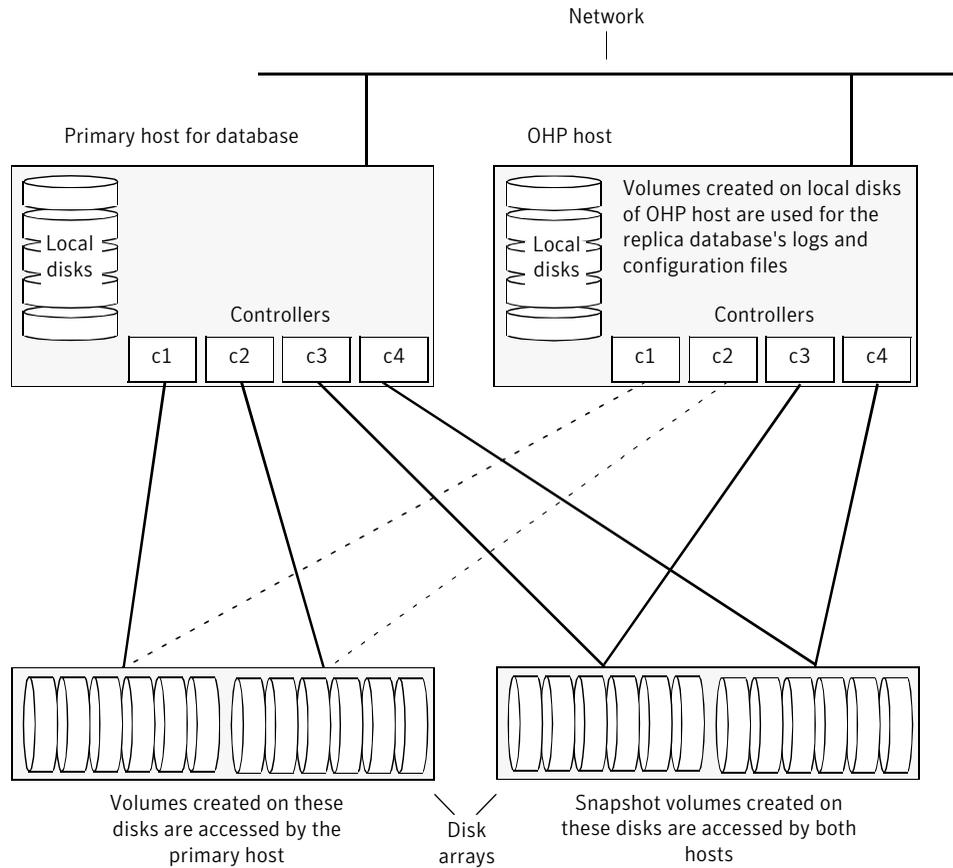
When you want to resynchronize a snapshot with the primary database, shut down the replica database, unmount the snapshot volume, and go back to step 3 to refresh the contents of the snapshot from the original volume.

Creating an off-host replica database

Figure 15-2 shows an example where the primary database volumes to be backed up, `dbase_vol` and `dbase_logs`, are configured on disks attached to controllers `c1` and `c2`, and the snapshots are to be created on disks attached to controllers `c3` and `c4`.

There is no requirement for the OHP host to have access to the disks that contain the primary database volumes.

Figure 15-2 Example system configuration for off-host decision support



Note: It is assumed that you have already prepared the database volumes to be replicated as described in the example.

See [“About setting up volumes for instant snapshots”](#) on page 141.

To set up a replica database to be used for decision support on an OHP host

- 1 If you have not already done so, prepare the OHP host to use the snapshot volume that contains the copy of the database tables. Set up any new database logs and configuration files that are required to initialize the database.
See [“About preparing a replica Oracle database”](#) on page 521.
- 2 On the primary host, use the following command to make a full-sized snapshot, `snapvol`, of the tablespace volume by breaking off plexes from the original volume:

```
# vxsnap -g volumedg make \  
  source=volume/newvol=snapvol/nmirror=N
```

The `nmirror` attribute specifies the number of mirrors, `N`, in the snapshot volume.

If the volume does not have any available plexes, or its layout does not support plex break-off, prepare an empty volume for the snapshot.

See [“Creating a volume for use as a full-sized instant snapshot”](#) on page 150.

Then use the following command to create the snapshot:

```
# vxsnap -g volumedg make source=volume/snapvol=snapvol
```

If a database spans more than one volume, specify all the volumes and their snapshot volumes as separate tuples on the same line.

For example:

```
# vxsnap -g dbasedg make source=vol1/snapvol=svol1 \  
  source=vol2/snapvol=svol2 source=vol3/snapvol=svol3
```

Note: This step sets up the snapshot volumes, and starts tracking changes to the original volumes.

When you are ready to create the replica database, proceed to step 3.

- 3 If the volumes to be backed up contain database tables in file systems, suspend updates to the volumes:
 - DB2 provides the `write suspend` command to temporarily suspend I/O activity for a database. As the DB2 database administrator, use a script such as the example script.
See [“Script to suspend I/O for a DB2 database”](#) on page 512.

Note that if the replica database must be able to be rolled forward (for example, if it is to be used as a standby database), the primary database must be in LOGRETAIN RECOVERY mode.

- Oracle supports online backup by temporarily suspending updates to the datafiles of the tablespaces, provided that the database is running in archive mode and the tablespaces are online. As the Oracle database administrator, put each tablespace into hot backup mode using a script such as the example script.

See “[Script to put an Oracle database into hot backup mode](#)” on page 511.

- Sybase ASE from version 12.0 onward provides the Quiesce feature to allow temporary suspension of writes to a database. As the Sybase database administrator, put the database in quiesce mode by using a script such as the example script.

See “[Script to quiesce a Sybase ASE database](#)” on page 512.

If you are using Sybase ASE 12.5, you can specify the `for external dump` clause to the `quiesce` command. This warm standby method allows you to update a replica database using transaction logs dumped from the primary database.

See “[Updating a warm standby Sybase ASE 12.5 database](#)” on page 192.

- 4 On the primary host, refresh the contents of the snapshot volumes from the original volume using the following command:

```
# vxsnap -g volumedg refresh snapvol source=vol \
  [snapvol2 source=vol2]... syncing=yes
```

The `syncing=yes` attribute starts a synchronization of the snapshot in the background.

For example, to refresh the snapshots `svol1`, `svol2` and `svol3`:

```
# vxsnap -g dbasedg refresh svol1 source=vol1 \
  svol2 source=vol2 svol3 source=vol3
```

- 5 If you temporarily suspended updates to volumes in 2, release all the tablespaces or databases from suspend, hot backup or quiesce mode:
 - As the DB2 database administrator, use a script such as that shown in the example.
See “[Script to resume I/O for a DB2 database](#)” on page 514.
 - As the Oracle database administrator, release all the tablespaces from hot backup mode using a script such as that shown in the example.
See “[Script to end Oracle database hot backup mode](#)” on page 513.

- As the Sybase database administrator, release the database from quiesce mode using a script such as that shown in the example. See “[Script to release a Sybase ASE database from quiesce mode](#)” on page 513.
- 6 Use the following command to wait for the contents of the snapshot to be fully synchronous with the contents of the original volume:

```
# vxsnap -g volumedg syncwait snapvol
```

For example, to wait for synchronization to finish for all the snapshots *svol1*, *svol2* and *svol3*, you would issue three separate commands:

```
# vxsnap -g dbasedg syncwait svol1
# vxsnap -g dbasedg syncwait svol2
# vxsnap -g dbasedg syncwait svol3
```

Note: You cannot move a snapshot volume into a different disk group until synchronization of its contents is complete. You can use the `vxsnap print` command to check on the progress of synchronization.

- 7 On the primary host, use the following command to split the disks containing the snapshot volumes into a separate disk group, *snapvoldg*, from the original disk group, *volumedg*:

```
# vxdg split volumedg
    snapvoldg
    snapvol ...
```

For example to split the snap volumes from *dbasedg*:

```
# vxdg split dbasedg snapvoldg svol1 svol2 svol3
```

- 8 On the primary host, deport the snapshot volume’s disk group using the following command:

```
# vxdg deport snapvoldg
```

- 9 On the OHP host where the replica database is to be set up, use the following command to import the snapshot volume’s disk group:

```
# vxdg import snapvoldg
```

- 10 VxVM will recover the volumes automatically after the disk group import unless it is set not to recover automatically. Check if the snapshot volume is initially disabled and not recovered following the split.

If a volume is in the DISABLED state, use the following command on the OHP host to recover and restart the snapshot volume:

```
# vxrecover -g snapvoldg -m snapvol ...
```

- 11 On the OHP host, for each snapshot volume containing tablespaces, check the file system that it contains, and mount the volume using the following commands:

```
# fsck -V vxfs /dev/vx/rdisk/diskgroup/snapvol
# mount -V vxfs /dev/vx/dsk/diskgroup/snapvol
    mount_point
```

For example, to check the file system in the snapshot volume `snap1_dbase_vol`, and mount it on `/rep/dbase_vol`:

```
# fsck -V vxfs /dev/vx/rdisk/snapvoldg/snap1_dbase_vol
# mount -V vxfs /dev/vx/dsk/snapvoldg/snap1_dbase_vol \
    /rep/dbase_vol
```

Note: For a replica DB2 database, the database volume must be mounted in the same location as on the primary host.

- 12 Copy any required log files from the primary host to the OHP host:
- For an Oracle database on the OHP host, copy the archived log files that were generated while the database was in hot backup mode to the new database's archived log directory (for example, `/rep/archlog`).
 - For a Sybase ASE database on the primary host, if you specified the `for external dump` clause when you quiesced the database, use the following `isql` command as the database administrator to dump the transaction log for the database:

```
dump transaction to dump_device with standby_access
```

Then copy the dumped transaction log to the appropriate database directory on the OHP host.

- 13 As the database administrator, start the new database:

- If the replica DB2 database is not to be rolled forward, use the following commands to start and recover it:

```
db2start  
db2inidb database as snapshot
```

If the replica DB2 database is to be rolled forward:

- The primary must have been placed in LOGRETAIN RECOVERY mode before the snapshot was taken.
- Use the following commands to start it, and put it in roll-forward pending state:

```
db2start  
db2inidb database as standby
```

Obtain the latest log files from the primary database, and use the following command to roll the replica database forward to the end of the logs:

```
db2 rollforward db database to end of logs
```

- For an Oracle database, use a script such as that shown in the example. See [“Script to complete, recover and start a replica Oracle database”](#) on page 517. (This script also creates the control file for the new database by executing the SQL script that you created using the procedure in the section preparing the replica Oracle database.) See [“About preparing a replica Oracle database”](#) on page 521.
- For a Sybase ASE database, use a script such as that shown in the example. See [“Script to start a replica Sybase ASE database”](#) on page 519. If you are using the warm standby method, specify the `-q` option to the `dataserver` command. Use the following `isql` commands to load the dump of the transaction log and put the database online:

```
load transaction from dump_device with standby_access  
online database database_name for standby_access
```

If you are not using the warm standby method, use the following `isql` command to recover the database, roll back any uncommitted transactions to the time that the quiesce command was issued, and put the database online:

```
online database database_name
```

Resynchronizing the data with the primary host

This procedure describes how to resynchronize the data in a snapshot with the primary host.

To resynchronize a snapshot with the primary database

- 1 On the OHP host, shut down the replica database, and use the following command to unmount each of the snapshot volumes:

```
# umount mount_point
```

- 2 On the OHP host, use the following command to deport the snapshot volume's disk group:

```
# vxdg deport snapvoldg
```

- 3 On the primary host, re-import the snapshot volume's disk group using the following command:

```
# vxdg [-s] import snapvoldg
```

Note: Specify the `-s` option if you are reimporting the disk group to be rejoined with a shared disk group in a cluster.

- 4 On the primary host, use the following command to rejoin the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join snapvoldg  
          volumedg
```

- 5 VxVM will recover the volumes automatically after the join unless it is set not to recover automatically. Check if the snapshot volumes are initially disabled and not recovered following the join.

If a volume is in the DISABLED state, use the following command on the primary host to recover and restart the snapshot volume:

```
# vxrecover -g volumedg -m snapvol
```

- 6 Use the steps in [Creating an off-host replica database](#) to resynchronize the snapshot and make the snapshot available at OHP host again.

The snapshots are now ready to be re-used for backup or for other decision support applications.

Updating a warm standby Sybase ASE 12.5 database

If you specified the `for external dump` clause when you quiesced the primary database, and you started the replica database by specifying the `-q` option to the `dataserver` command, you can use transaction logs to update the replica database.

To update the replica database

- 1 On the primary host, use the following `isql` command to dump the transaction log for the database:

```
dump transaction to dump_device with standby_access
```

Copy the transaction log dump to the appropriate database directory on the OHP host.

- 2 On the OHP host, use the following `isql` command to load the new transaction log:

```
load transaction from dump_device with standby_access
```

- 3 On the OHP host, use the following `isql` command to put the database online:

```
online database database_name for standby_access
```

Reattaching snapshot plexes

Some or all plexes of an instant snapshot may be reattached to the specified original volume, or to a source volume in the snapshot hierarchy above the snapshot volume.

Note: This operation is not supported for space-optimized instant snapshots.

By default, all the plexes are reattached, which results in the removal of the snapshot. If required, the number of plexes to be reattached may be specified as the value assigned to the `nmirror` attribute.

Note: The snapshot being reattached must not be open to any application. For example, any file system configured on the snapshot volume must first be unmounted.

To reattach a snapshot

- ◆ Use the following command, to re-attach some or all plexes of an instant snapshot to the specified original volume, or to a source volume in the snapshot hierarchy above the snapshot volume:

```
# vxsnap [-g diskgroup] reattach snapvol source=vol \  
[nmirror=number]
```

For example the following command reattaches 1 plex from the snapshot volume, `snapmyvol`, to the volume, `myvol`:

```
# vxsnap -g mydg reattach snapmyvol source=myvol nmirror=1
```

While the reattached plexes are being resynchronized from the data in the parent volume, they remain in the `SNAPTMP` state. After resynchronization is complete, the plexes are placed in the `SNAPDONE` state.

Database recovery

This chapter includes the following topics:

- [About database recovery using Storage Checkpoints](#)
- [Creating Storage Checkpoints](#)
- [Rolling back a database](#)

About database recovery using Storage Checkpoints

You can use Storage Checkpoints to implement efficient backup and recovery of databases that have been laid out on VxFS file systems. A Storage Checkpoint allows you to roll back an entire database, a tablespace, or a single database file to the time that the Storage Checkpoint was taken. Rolling back or restoring from any Storage Checkpoint is generally very fast because only the changed data blocks need to be restored.

Storage Checkpoints can also be mounted, allowing regular file system operations to be performed or secondary databases to be started.

This chapter provides an introduction to using Storage Checkpoints for Storage Rollback of an Oracle database. For full information on how to administer the Storage Checkpoints feature for the database software that you are using:

Note: Storage Checkpoints can only be used to restore from logical errors such as human mistakes or software faults. You cannot use them to restore files after a disk failure because all the data blocks are on the same physical device. Disk failure requires restoration of a database from a backup copy of the database files kept on a separate medium. Combining data redundancy (for example, disk mirroring) with Storage Checkpoints is recommended for highly critical data to protect against both physical media failure and logical errors.

Storage Checkpoints require space in the file systems where they are created, and the space required grows over time as copies of changed file system blocks are made. If a file system runs out of space, and there is no disk space into which the file system and any underlying volume can expand, VxFS automatically removes the oldest Storage Checkpoints if they were created with the removable attribute.

Creating Storage Checkpoints

To create Storage Checkpoints, select 3 Storage Checkpoint Administration > Create New Storage Checkpoints in the VxDBA utility. This can be done with a database either online or offline.

Note: To create a Storage Checkpoint while the database is online, ARCHIVELOG mode must be enabled in Oracle. During the creation of the Storage Checkpoint, the tablespaces are placed in backup mode. Because it only takes a few seconds to take a Storage Checkpoint, the extra redo logs generated while the tablespaces are in online backup mode are very small. To optimize recovery, it is recommended that you keep ARCHIVELOG mode enabled.

Warning: Changes to the structure of a database, such as the addition or removal of datafiles, make Storage Rollback impossible when made after a Storage Checkpoint is taken. A backup copy of the control file for the database is saved under the `/etc/vx/vxdba/ORACLE_SID/checkpoint_dir` directory immediately after a Storage Checkpoint is created. If necessary, you can use this file to assist with database recovery. If possible, both an ASCII and binary copy of the control file are made, with the binary version being compressed to conserve space. Use extreme caution if you attempt to recover your database using these control files. It is recommended that you remove old Storage Checkpoints and create new ones whenever you restructure a database.

Rolling back a database

The procedure in this section describes how to roll back a database using a Storage Checkpoint, for example, after a logical error has occurred.

To roll back a database

- 1 Ensure that the database is offline. You can use the VxDBA utility to display the status of the database and its tablespaces, and to shut down the database:
 - Select 2 Display Database/VxDBA Information to access the menus that display status information.

- Select 1 Database Administration > Shutdown Database Instance to shut down a database.
- 2 Select 4 Storage Rollback Administration > Roll Back the Database to a Storage Checkpoint in the VxDBA utility, and choose the appropriate Storage Checkpoint. This restores all data files used by the database, except redo logs and control files, to their state at the time that the Storage Checkpoint was made.
- 3 Start up, but do not open, the database instance by selecting 1 Database Administration > Startup Database Instance in the VxDBA utility.
- 4 Use one of the following commands to perform an incomplete media recovery of the database:

- Recover the database until you stop the recovery:

```
recover database until cancel;  
...  
alter database [database] recover cancel;
```

- Recover the database to the point just before a specified system change number, scn:

```
recover database until change scn;
```

- Recover the database to the specified time:

```
recover database until time 'yyyy-mm-dd:hh:mm:ss';
```

- Recover the database to the specified time using a backup control file:

```
recover database until time 'yyyy-mm-dd:hh:mm:ss' \  
using backup controlfile;
```

Note: To find out when an error occurred, check the `../bdump/alert*.log` file.

See the Oracle documentation for complete and detailed information on database recovery.

- 5 To open the database after an incomplete media recovery, use the following command:

```
alter database open resetlogs;
```

Note: The `resetlogs` option is required after an incomplete media recovery to reset the log sequence. Remember to perform a full database backup and create another Storage Checkpoint after log reset.

- 6 Perform a full database backup, and use the VxDBA utility to remove any existing Storage Checkpoints that were taken before the one to which you just rolled back the database. These Storage Checkpoints can no longer be used for Storage Rollback. If required, use the VxDBA utility to delete the old Storage Checkpoints and to create new ones.

Administering volume snapshots

This chapter includes the following topics:

- [About volume snapshots](#)
- [Traditional third-mirror break-off snapshots](#)
- [Full-sized instant snapshots](#)
- [Space-optimized instant snapshots](#)
- [Emulation of third-mirror break-off snapshots](#)
- [Linked break-off snapshot volumes](#)
- [Cascaded snapshots](#)
- [Creating multiple snapshots](#)
- [Restoring the original volume from a snapshot](#)
- [Creating instant snapshots](#)
- [Creating traditional third-mirror break-off snapshots](#)
- [Adding a version 0 DCO and DCO volume](#)

About volume snapshots

VxVM can take an image of a volume at a given point in time. This image is called a volume snapshot.

See [“Volume snapshots”](#) on page 134.

You can also take a snapshot of a volume set.

See [“Creating instant snapshots”](#) on page 212.

Volume snapshots allow you to make backup copies of your volumes online with minimal interruption to users. You can then use the backup copies to restore data that has been lost due to disk failure, software errors or human mistakes, or to create replica volumes for the purposes of report generation, application development, or testing.

Volume snapshots can also be used to implement off-host online backup.

See [“About off-host cluster file system backup”](#) on page 167.

A volume snapshot captures the data that exists in a volume at a given point in time. As such, VxVM does not have any knowledge of data that is cached in memory by the overlying file system, or by applications such as databases that have files open in the file system. Snapshots are always crash consistent, that is, they can be put to use by letting the application perform its recovery. This is similar to how the application recovery occurs after a server crash. If the `ESgen` volume usage type is set on a volume that contains a mounted Veritas File System (VxFS), VxVM coordinates with VxFS to flush data that is in the cache to the volume. For other file system types, depending on the capabilities of the file system, there may potentially be inconsistencies between data in memory and in the snapshot.

For databases, a suitable mechanism must additionally be used to ensure the integrity of tablespace data when the volume snapshot is taken. The facility to temporarily suspend file system I/O is provided by most modern database software. For ordinary files in a file system, which may be open to a wide variety of different applications, there may be no way to ensure the complete integrity of the file data other than by shutting down the applications and temporarily unmounting the file system. In many cases, it may only be important to ensure the integrity of file data that is not in active use at the time that you take the snapshot.

There are two alternative methods of creating volume snapshots:

- Creating instant snapshots
See [“Creating instant snapshots”](#) on page 212.
- Creating traditional third-mirror break-off snapshots
See [“Creating traditional third-mirror break-off snapshots”](#) on page 244.

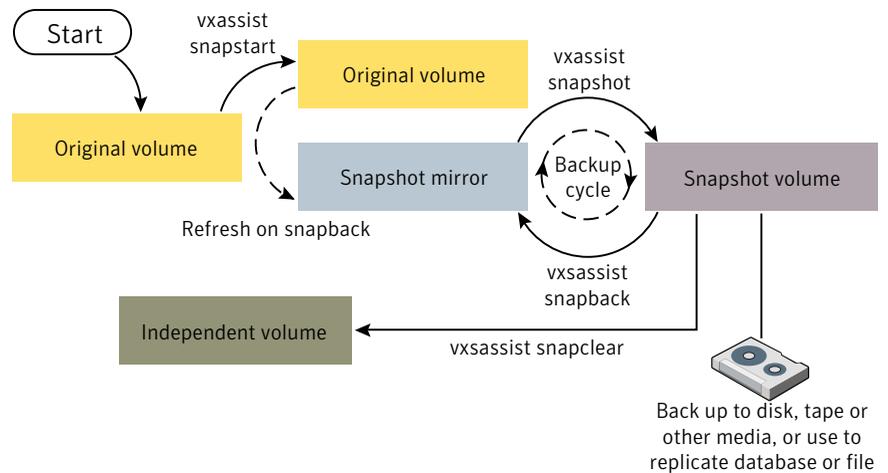
Snapshot creation using the `vxsnap` command is the preferred mechanism for implementing point-in-time copy solutions in VxVM. Support for traditional third-mirror snapshots that are created using the `vxassist` command may be removed in a future release.

To recover from the failure of instant snapshot commands, see the *Veritas Volume Manager Troubleshooting Guide*.

Traditional third-mirror break-off snapshots

Figure 17-1 shows the traditional third-mirror break-off volume snapshot model that is supported by the `vxassist` command.

Figure 17-1 Third-mirror snapshot creation and usage



The `vxassist snapstart` command creates a mirror to be used for the snapshot, and attaches it to the volume as a snapshot mirror. As is usual when creating a mirror, the process of copying the volume’s contents to the new snapshot plexes can take some time to complete. (The `vxassist snapabort` command cancels this operation and removes the snapshot mirror.)

See “Full-sized instant snapshots” on page 202.

See “Space-optimized instant snapshots” on page 204.

When the attachment is complete, the `vxassist snapshot` command is used to create a new snapshot volume by taking one or more snapshot mirrors to use as its data plexes. The snapshot volume contains a copy of the original volume’s data at the time that you took the snapshot. If more than one snapshot mirror is used, the snapshot volume is itself mirrored.

The command, `vxassist snapback`, can be used to return snapshot plexes to the original volume from which they were snapped, and to resynchronize the data in the snapshot mirrors from the data in the original volume. This enables you to refresh the data in a snapshot after you use it to make a backup. You can use a variation of the same command to restore the contents of the original volume from a snapshot previously taken.

The FastResync feature minimizes the time and I/O needed to resynchronize the data in the snapshot. If FastResync is not enabled, a full resynchronization of the data is required. For more details:

See the *Veritas Volume Manager Administrator's Guide*.

Finally, you can use the `vxassist snapclear` command to break the association between the original volume and the snapshot volume. Because the snapshot relationship is broken, no change tracking occurs. Use this command if you do not need to reuse the snapshot volume to create a new point-in-time copy.

The use of the `vxassist` command to administer traditional (third-mirror break-off) snapshots is not supported for volumes that are prepared for instant snapshot creation. Use the `vxsnap` command instead.

See “Full-sized instant snapshots” on page 202.

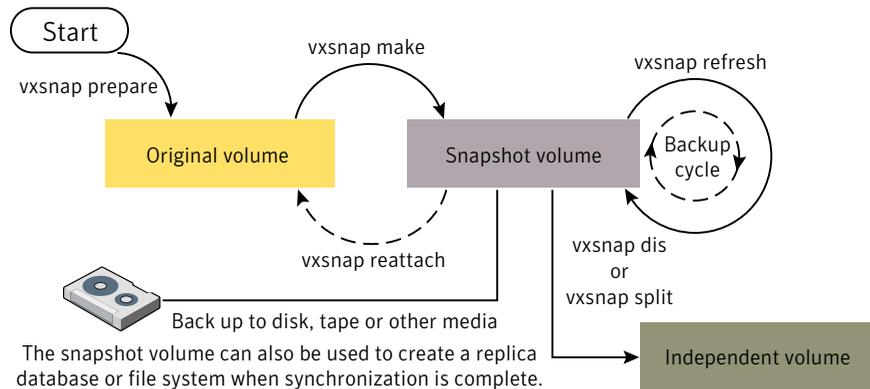
See “Space-optimized instant snapshots” on page 204.

Full-sized instant snapshots

Full-sized instant snapshots are a variation on the third-mirror volume snapshot model that make a snapshot volume available for I/O access as soon as the snapshot plexes have been created.

Figure 17-2 shows the full-sized instant volume snapshot model.

Figure 17-2 Full-sized instant snapshot creation and usage in a backup cycle



To create an instant snapshot, use the `vxsnap make` command. This command can either be applied to a suitably prepared empty volume that is to be used as the snapshot volume, or it can be used to break off one or more synchronized

plexes from the original volume (which is similar to the way that the `vxassist` command creates its snapshots).

Unlike a third-mirror break-off snapshot created using the `vxassist` command, you can make a backup of a full-sized instant snapshot, instantly refresh its contents from the original volume, or attach its plexes to the original volume, without completely synchronizing the snapshot plexes from the original volume.

VxVM uses a copy-on-write mechanism to ensure that the snapshot volume preserves the contents of the original volume at the time that the snapshot is taken. Any time that the original contents of the volume are about to be overwritten, the original data in the volume is moved to the snapshot volume before the write proceeds. As time goes by, and the contents of the volume are updated, its original contents are gradually relocated to the snapshot volume.

If a read request comes to the snapshot volume, yet the data resides on the original volume (because it has not yet been changed), VxVM automatically and transparently reads the data from the original volume.

If desired, you can perform either a background (non-blocking) or foreground (blocking) synchronization of the snapshot volume. This is useful if you intend to move the snapshot volume into a separate disk group for off-host processing, or you want to turn the snapshot volume into an independent volume.

The `vxsnap refresh` command allows you to update the data in a snapshot, for example, before taking a backup.

The command `vxsnap reattach` attaches snapshot plexes to the original volume, and resynchronizes the data in these plexes from the original volume. Alternatively, you can use the `vxsnap restore` command to restore the contents of the original volume from a snapshot that you took at an earlier point in time. You can also choose whether or not to keep the snapshot volume after restoration of the original volume is complete.

By default, the FastResync feature of VxVM is used to minimize the time and I/O needed to resynchronize the data in the snapshot mirror. FastResync must be enabled to create instant snapshots.

For more information on FastResync:

See the *Veritas Volume Manager Administrator's Guide*.

See [“Creating and managing full-sized instant snapshots”](#) on page 221.

An empty volume must be prepared for use by full-sized instant snapshots and linked break-off snapshots.

See [“Creating a volume for use as a full-sized instant or linked break-off snapshot”](#) on page 217.

Space-optimized instant snapshots

Volume snapshots require a complete copy of the original volume, and use as much storage space as the original volume.

In contrast, space-optimized instant snapshots do not require a complete copy of the original volume’s storage space. They use a storage cache.

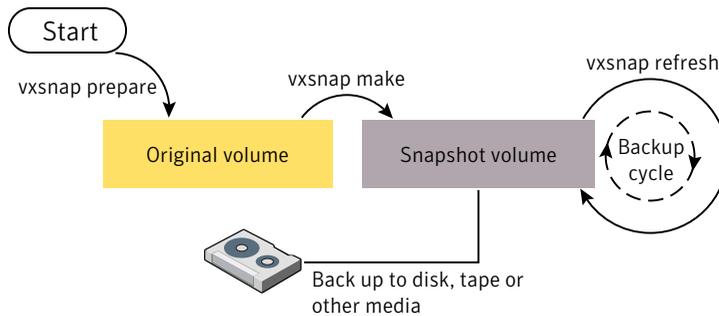
You may find it convenient to configure a single storage cache in a disk group that can be shared by all the volumes in that disk group. If so, the name of the cache that is declared must be the same for each volume’s space-optimized snapshot. The cache is stored on disk and is persistent.

See [“Creating a shared cache object”](#) on page 151.

When the original volume is written to, VxVM preserves the original data contents in the cache before the write is committed. As the storage cache typically requires much less storage than the original volume, it is referred to as space-optimized. If the cache approaches full, you can configure VxVM to increase the cache automatically using any available free space in the disk group.

[Figure 17-3](#) shows the instant space-optimized snapshot model.

Figure 17-3 Space-optimized instant snapshot creation and usage in a backup cycle



Space-optimized snapshots use a copy-on-write mechanism to make them immediately available for use when they are first created, or when their data is refreshed. Unlike instant snapshots, you cannot enable synchronization on space-optimized snapshots, reattach them to their original volume, or turn them into independent volumes.

See [“Creating and managing space-optimized instant snapshots”](#) on page 218.

A cache object and cache volume must be set up for use by space-optimized instant snapshots.

See [“Creating a shared cache object”](#) on page 151.

Emulation of third-mirror break-off snapshots

Third-mirror break-off snapshots are suitable for write-intensive volumes (such as for database redo logs) where the copy-on-write mechanism of space-optimized or full-sized instant snapshots might degrade performance.

If you use the `vxsnap prepare` command to enable a volume for use with instant and space-optimized snapshots, you cannot use the `vxassist` snapshot commands to administer snapshots that you create for that volume. If you require snapshots that behave as third-mirror break-off snapshots (that is, they must be fully synchronized before they can be used), there are three ways to achieve this:

- Use the `vxsnap addmir` command to create and attach one or more snapshot mirrors to the volume. When the plexes have been synchronized and are in the `SNAPDONE` state, the `vxsnap make` command can then be used with the `nmirror` attribute to create the snapshot volume. This technique is similar to using the `vxassist snapstart` and `vxassist snapshot` commands. See [“Traditional third-mirror break-off snapshots”](#) on page 201.

- Use the `vxsnap make` command with the `plex` attribute to use one or more existing plexes of a volume as snapshot plexes. The volume must have a sufficient number of available plexes that are in the `ACTIVE` state.

The volume must be a non-layered volume with a `mirror` or `mirror-stripe` layout, or a RAID-5 volume that you have converted to a special layered volume and then mirrored.

The plexes in a volume with a `stripe-mirror` layout are mirrored at the sub-volume level, and cannot be used for snapshots.

- Use the `vxsnap make` command with the `sync=yes` and `type=full` attributes specified to create the snapshot volume, and then use the `vxsnap syncwait` command to wait for synchronization of the snapshot volume to complete.

See [“Adding snapshot mirrors to a volume”](#) on page 231.

See [“Creating and managing third-mirror break-off snapshots”](#) on page 223.

Linked break-off snapshot volumes

A variant of third-mirror break-off snapshots are linked break-off snapshots, which use the `vxsnap addmir` command to link a specially prepared volume with the data volume. The volume that is used for the snapshot is prepared in the same way as for full-sized instant snapshots. However, unlike full-sized instant snapshots, this volume can be set up in a different disk group from the data volume. This makes linked break-off snapshots especially suitable for recurring off-host processing applications as it avoids the disk group split/join administrative step.

As with third-mirror break-off snapshots, you must wait for the contents of the snapshot volume to be synchronized with the data volume before you can use the `vxsnap make` command to take the snapshot.

When a link is created between a volume and the mirror that will become the snapshot, separate link objects (similar to snap objects) are associated with the volume and with its mirror. The link object for the original volume points to the mirror volume, and the link object for the mirror volume points to the original volume. All I/O is directed to both the original volume and its mirror, and a synchronization of the mirror from the data in the original volume is started.

You can use the `vxprint` command to display the state of link objects, which appear as type `ln`. Link objects can have the following states:

ACTIVE	The mirror volume has been fully synchronized from the original volume. The <code>vxsnap make</code> command can be run to create a snapshot.
ATTACHING	Synchronization of the mirror volume is in progress. The <code>vxsnap make</code> command cannot be used to create a snapshot until the state changes to ACTIVE. The <code>vxsnap snapwait</code> command can be used to wait for the synchronization to complete.
BROKEN	The mirror volume has been detached from the original volume because of an I/O error or an unsuccessful attempt to grow the mirror volume. The <code>vxrecover</code> command can be used to recover the mirror volume in the same way as for a DISABLED volume.

If you resize (grow or shrink) a volume, all its ACTIVE linked mirror volumes are also resized at the same time. The volume and its mirrors can be in the same disk group or in different disk groups. If the operation is successful, the volume and its mirrors will have the same size.

If a volume has been grown, a resynchronization of the grown regions in its linked mirror volumes is started, and the links remain in the ATTACHING state until resynchronization is complete. The `vxsnap snapwait` command can be used to wait for the state to become ACTIVE.

When you use the `vxsnap make` command to create the snapshot volume, this removes the link, and establishes a snapshot relationship between the snapshot volume and the original volume.

The `vxsnap reattach` operation re-establishes the link relationship between the two volumes, and starts a resynchronization of the mirror volume.

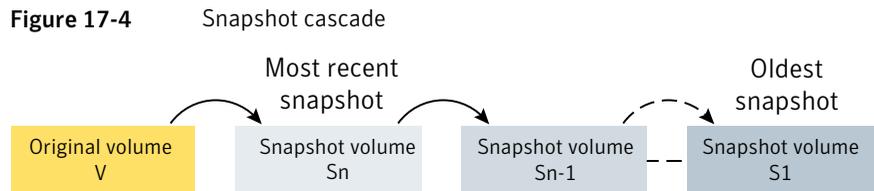
See [“Creating and managing linked break-off snapshot volumes”](#) on page 226.

An empty volume must be prepared for use by linked break-off snapshots.

See “Creating a volume for use as a full-sized instant or linked break-off snapshot” on page 217.

Cascaded snapshots

Figure 17-4 shows a snapshot hierarchy, known as a snapshot cascade, that can improve write performance for some applications:



Instead of having several independent snapshots of the volume, it is more efficient to make the older snapshots into children of the latest snapshot.

A snapshot cascade is most likely to be used for regular online backup of a volume where space-optimized snapshots are written to disk but not to tape.

A snapshot cascade improves write performance over the alternative of several independent snapshots, and also requires less disk space if the snapshots are space-optimized. Only the latest snapshot needs to be updated when the original volume is updated. If and when required, the older snapshots can obtain the changed data from the most recent snapshot.

A snapshot may be added to a cascade by specifying the `infrontof` attribute to the `vxsnap make` command when the second and subsequent snapshots in the cascade are created. Changes to blocks in the original volume are only written to the most recently created snapshot volume in the cascade. If an attempt is made to read data from an older snapshot that does not exist in that snapshot, it is obtained by searching recursively up the hierarchy of more recent snapshots.

The following points determine whether it is appropriate to use a snapshot cascade:

- The deletion of a snapshot in the cascade copies the snapshot’s data to the next snapshot in the cascade.
- The reliability of a snapshot in the cascade depends on all the newer snapshots in the chain. Thus the oldest snapshot in the cascade is the most vulnerable.
- Reading from a snapshot in the cascade may require data to be fetched from one or more other snapshots in the cascade.

For these reasons, it is recommended that you do not attempt to use a snapshot cascade with applications that need to remove or split snapshots from the cascade.

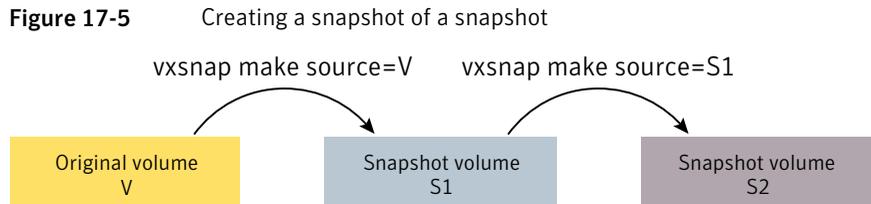
In such cases, it may be more appropriate to create a snapshot of a snapshot as described in the following section.

See “[Adding a snapshot to a cascaded snapshot hierarchy](#)” on page 232.

Note: Only unsynchronized full-sized or space-optimized instant snapshots are usually cascaded. It is of little use to create cascaded snapshots if the `infrontof` snapshot volume is fully synchronized (as, for example, with break-off type snapshots).

Creating a snapshot of a snapshot

[Figure 17-5](#) creation of a snapshot of an existing snapshot.



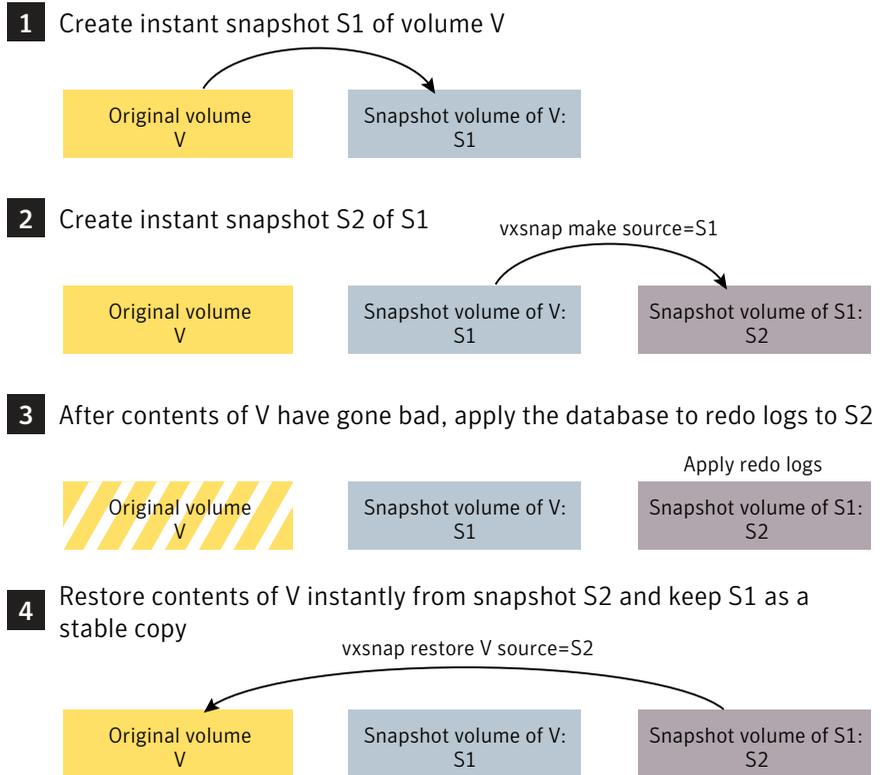
Even though the arrangement of the snapshots in this figure appears similar to a snapshot cascade, the relationship between the snapshots is not recursive. When reading from the snapshot `s2`, data is obtained directly from the original volume, `v`, if it does not exist in `s1` itself.

See [Figure 17-4](#) on page 207.

Such an arrangement may be useful if the snapshot volume, `s1`, is critical to the operation. For example, `s1` could be used as a stable copy of the original volume, `v`. The additional snapshot volume, `s2`, can be used to restore the original volume if that volume becomes corrupted. For a database, you might need to replay a redo log on `s2` before you could use it to restore `v`.

[Figure 17-6](#) shows the sequence of steps that would be required to restore a database.

Figure 17-6 Using a snapshot of a snapshot to restore a database



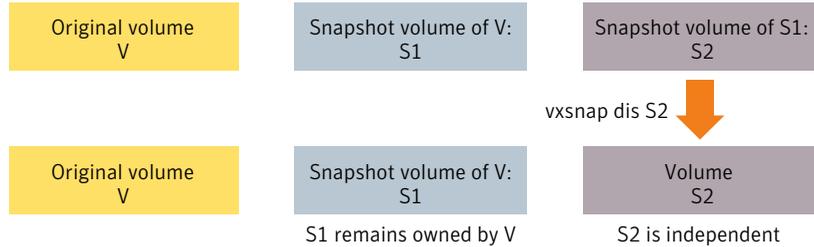
If you have configured snapshots in this way, you may wish to make one or more of the snapshots into independent volumes. There are two `vxsnap` commands that you can use to do this:

- `vxsnap dis` dissociates a snapshot and turns it into an independent volume. The snapshot to be dissociated must have been fully synchronized from its parent. If a snapshot volume has a child snapshot volume, the child must also have been fully synchronized. If the command succeeds, the child snapshot becomes a snapshot of the original volume.

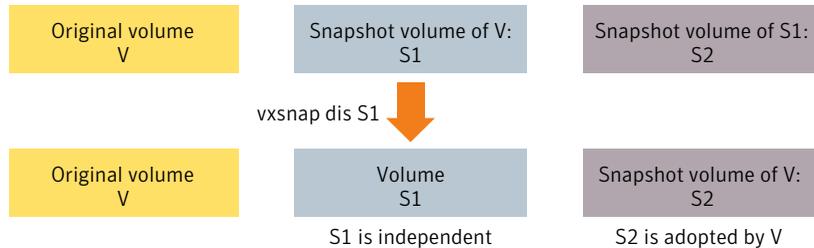
Figure 17-7 shows the effect of applying the `vxsnap dis` command to snapshots with and without dependent snapshots.

Figure 17-7 Dissociating a snapshot volume

`vxsnap dis` is applied to snapshot S2, which has no snapshots of its own



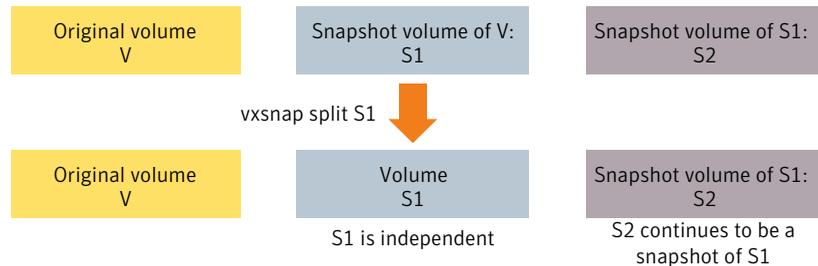
`vxsnap dis` is applied to snapshot S1, which has one snapshot S2



- `vxsnap split` dissociates a snapshot and its dependent snapshots from its parent volume. The snapshot that is to be split must have been fully synchronized from its parent volume.

Figure 17-8 shows the operation of the `vxsnap split` command.

Figure 17-8 Splitting snapshots



Creating multiple snapshots

To make it easier to create snapshots of several volumes at the same time, both the `vxsnap make` and `vxassist snapshot` commands accept more than one volume name as their argument.

For traditional snapshots, you can create snapshots of all the volumes in a single disk group by specifying the option `-o allvols` to the `vxassist snapshot` command.

By default, each replica volume is named `SNAPnumber-volume`, where *number* is a unique serial number, and *volume* is the name of the volume for which a snapshot is being taken. This default can be overridden by using the option `-o name=pattern`.

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

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

You can create a snapshot of all volumes that form a logical group; for example, all the volumes that conform to a database instance.

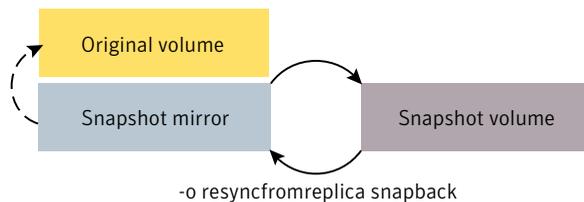
Restoring the original volume from a snapshot

For traditional snapshots, the snapshot plex is resynchronized from the data in the original volume during a `vxassist snapback` operation.

[Figure 17-9](#) shows an alternative where the snapshot overwrites the original volume.

Figure 17-9 Resynchronizing an original volume from a snapshot

Refresh on snapback



Specifying the option `-o resyncfromreplica` to `vxassist resync` resynchronizes the original volume from the data in the snapshot.

Warning: The original volume must not be in use during a `snapback` operation that specifies the option `-o resyncfromreplica` to resynchronize the volume from a snapshot. Stop any application, such as a database, and unmount any file systems that are configured to use the volume.

For instant snapshots, the `vxsnap restore` command may be used to restore the contents of the original volume from an instant snapshot or from a volume derived

from an instant snapshot. The volume that is used to restore the original volume can either be a true backup of the original's contents, or it may have been modified in some way (for example, by applying a database log replay or by running a file system checking utility such as `fsck`). All synchronization of the contents of this backup must be completed before the original volume can be restored from it. The original volume is immediately available for use while its contents are being restored.

See “[Restoring a volume from an instant snapshot](#)” on page 235.

Creating instant snapshots

VxVM allows you to make instant snapshots by using the `vxsnap` command.

You can also take instant snapshots of RAID-5 volumes that have been converted to a special layered volume layout by the addition of a DCO and DCO volume.

A plex in a full-sized instant snapshot requires as much space as the original volume. If you instead make a space-optimized instant snapshot of a volume, this only requires enough storage to record the original contents of the parent volume as they are changed during the life of the snapshot.

The recommended approach to performing volume backup from the command line, or from a script, is to use the `vxsnap` command. The `vxsnap prepare` and `make` tasks allow you to back up volumes online with minimal disruption to users.

`vxsnap prepare` creates a DCO and DCO volume and associates this with the original volume. It also enables Persistent FastResync.

`vxsnap make` creates an instant snapshot that is immediately available for making a backup. After the snapshot has been taken, read requests for data in the instant snapshot volume are satisfied by reading either from a non-updated region of the original volume, or from the copy of the original contents of an updated region that have been recorded by the snapshot.

Note: Synchronization of a full-sized instant snapshot from the original volume is enabled by default. If you specify the `syncing=no` attribute to `vxsnap make`, this disables synchronization, and the contents of the instant snapshot are unlikely ever to become fully synchronized with those of the original volume at the point in time that the snapshot was taken. In such a case, the snapshot cannot be used for off-host processing, nor can it become an independent volume.

You can immediately retake a full-sized or space-optimized instant snapshot at any time by using the `vxsnap refresh` command. If a fully synchronized instant snapshot is required, the new resynchronization must first complete.

To create instant snapshots of volume sets, use volume set names in place of volume names in the `vxsnap` command.

See [“Creating instant snapshots of volume sets”](#) on page 229.

When using the `vxsnap prepare` or `vxassist make` commands to make a volume ready for instant snapshot operations, if the specified region size exceeds half the value of the tunable `voliomem_maxpool_sz`, the operation succeeds but gives a warning such as the following (for a system where `voliomem_maxpool_sz` is set to 12MB):

```
VxVM vxassist WARNING V-5-1-0 Specified regionsize is
larger than the limit on the system
(voliomem_maxpool_sz/2=6144k).
```

If this message is displayed, `vxsnap make`, `refresh` and `restore` operations on such volumes fail as they might potentially hang the system. Such volumes can be used only for break-off snapshot operations using the `reattach` and `make` operations.

To make the volumes usable for instant snapshot operations, use `vxsnap unprepare` on the volume, and then use `vxsnap prepare` to re-prepare the volume with a region size that is less than half the size of `voliomem_maxpool_sz` (in this example, 1MB):

```
# vxsnap -g mydg -f unprepare voll
# vxsnap -g mydg prepare voll regionsize=1M
```

See [“Creating instant snapshots of volume sets”](#) on page 229.

See [“Creating and managing space-optimized instant snapshots”](#) on page 218.

See [“Creating and managing full-sized instant snapshots”](#) on page 221.

See [“Creating and managing third-mirror break-off snapshots”](#) on page 223.

See [“Creating and managing linked break-off snapshot volumes”](#) on page 226.

Preparing to create instant and break-off snapshots

To prepare a volume for the creation of instant and break-off snapshots

- 1 Use the following commands to see if the volume has a version 20 data change object (DCO) and DCO volume that allow instant snapshots and Persistent FastResync to be used with the volume, and to check that FastResync is enabled on the volume:

```
# vxprint -g volmedg -F%instant volume  
# vxprint -g volmedg -F%fastresync volume
```

If both commands return a value of `on`, skip to step [v450228343](#). Otherwise continue with step [2](#).

- 2 To prepare a volume for instant snapshots, use the following command:

```
# vxsnap [-g diskgroup] prepare volume [regionsize=size] \  
  [ndcomirs=number] [alloc=storage_attributes]
```

Run the `vxsnap prepare` command on a volume only if it does not have a version 20 DCO volume (for example, if you have run the `vxsnap unprepare` command on the volume).

For example, to prepare the volume, `myvol`, in the disk group, `mydg`, use the following command:

```
# vxsnap -g mydg prepare myvol regionsize=128k ndcomirs=2 \  
  alloc=mydg10,mydg11
```

This example creates a DCO object and redundant DCO volume with two plexes located on disks `mydg10` and `mydg11`, and associates them with `myvol`. The region size is also increased to 128KB from the default size of 64KB. The region size must be a power of 2, and be greater than or equal to 16KB. A smaller value requires more disk space for the change maps, but the finer granularity provides faster resynchronization.

- 3 If you need several space-optimized instant snapshots for the volumes in a disk group, you may find it convenient to create a single shared cache object in the disk group rather than a separate cache object for each snapshot.

See [“Creating a shared cache object”](#) on page 215.

For full-sized instant snapshots and linked break-off snapshots, you must prepare a volume that is to be used as the snapshot volume. This volume must be the same size as the data volume for which the snapshot is being created, and it must also have the same region size.

See [“Creating a volume for use as a full-sized instant or linked break-off snapshot”](#) on page 217.

Creating a shared cache object

To create a shared cache object

- 1 Decide on the following characteristics that you want to allocate to the cache volume that underlies the cache object:
 - The cache volume size should be sufficient to record changes to the parent volumes during the interval between snapshot refreshes. A suggested value is 10% of the total size of the parent volumes for a refresh interval of 24 hours.
 - The cache volume can be mirrored for redundancy.
 - If the cache volume is mirrored, space is required on at least as many disks as it has mirrors. These disks should not be shared with the disks used for the parent volumes. The disks should not be shared with disks used by critical volumes to avoid impacting I/O performance for critical volumes, or hindering disk group split and join operations.
- 2 Having decided on its characteristics, use the `vxassist` command to create the cache volume. The following example creates a mirrored cache volume, `cachevol`, with size 1GB in the disk group, `mydg`, on the disks `mydg16` and `mydg17`:

```
# vxassist -g mydg make cachevol 1g layout=mirror \  
init=active mydg16 mydg17
```

The attribute `init=active` makes the cache volume immediately available for use.

- 3 Use the `vxmake cache` command to create a cache object on top of the cache volume that you created in the previous step:

```
# vxmake [-g diskgroup] cache cache_object \  
  cachevolname=volume [regionsize=size] [autogrow=on] \  
  [highwatermark=hwmk] [autogrowby=agbvalue] \  
  [maxautogrow=maxagbvalue]
```

If the region size, `regionsize`, is specified, it must be a power of 2, and be greater than or equal to 16KB (16k). If not specified, the region size of the cache is set to 64KB.

All space-optimized snapshots that share the cache must have a region size that is equal to or an integer multiple of the region size set on the cache. Snapshot creation also fails if the original volume's region size is smaller than the cache's region size.

If the region size of a space-optimized snapshot differs from the region size of the cache, this can degrade the system's performance compared to the case where the region sizes are the same.

To prevent the cache from growing automatically, specify `autogrow=off`. By default it is `autogrow=on`.

In the following example, the cache object, `cobjmydg`, is created over the cache volume, `cachevol`, the region size of the cache is set to 32KB, and the `autogrow` feature is enabled:

```
# vxmake -g mydg cache cobjmydg cachevolname=cachevol \  
  regionsize=32k autogrow=on
```

- 4 Enable the cache object using the following command:

```
# vxcache [-g diskgroup] start cache_object
```

For example to start the cache object, `cobjmydg`:

```
# vxcache -g mydg start cobjmydg
```

See [“Removing a cache”](#) on page 244.

Creating a volume for use as a full-sized instant or linked break-off snapshot

To create an empty volume for use by a full-sized instant snapshot or a linked break-off snapshot

- 1 Use the `vxprint` command on the original volume to find the required size for the snapshot volume.

```
# LEN=`vxprint [-g diskgroup] -F%len volume`
```

The command as shown assumes a Bourne-type shell such as `sh`, `ksh` or `bash`. You may need to modify the command for other shells such as `csh` or `tcsh`.

- 2 Use the `vxprint` command on the original volume to discover the name of its DCO:

```
# DCONAME=`vxprint [-g diskgroup] -F%dco_name volume`
```

- 3 Use the `vxprint` command on the DCO to discover its region size (in blocks):

```
# RSZ=`vxprint [-g diskgroup] -F%regionsz $DCONAME`
```

- 4 Use the `vxassist` command to create a volume, *snapvol*, of the required size and redundancy, together with a version 20 DCO volume with the correct region size:

```
# vxassist [-g diskgroup] make snapvol $LEN \  
[layout=mirror nmirror=number] logtype=dco dnl=off \  
dcoversion=20 [ndcomirror=number] regionsz=$RSZ \  
init=active [storage_attributes]
```

Storage attributes give you control over the devices, including disks and controllers, which `vxassist` uses to configure a volume.

Specify the same number of DCO mirrors (`ndcomirror`) as the number of mirrors in the volume (`nmirror`). The `init=active` attribute makes the volume available immediately. You can use storage attributes to specify which disks should be used for the volume.

As an alternative to creating the snapshot volume and its DCO volume in a single step, you can first create the volume, and then prepare it for instant snapshot operations as shown here:

```
# vxassist [-g diskgroup] make snapvol $LEN \  
[layout=mirror nmirror=number] init=active \  
[storage_attributes] \  
# vxsnap [-g diskgroup] prepare snapvol [ndcomirs=number] \  
regionsize=$RSZ [storage_attributes]
```

Creating and managing space-optimized instant snapshots

Space-optimized instant snapshots are not suitable for write-intensive volumes (such as for database redo logs) because the copy-on-write mechanism may degrade performance.

To split the volume and snapshot into separate disk groups (for example, to perform off-host processing), you must use a fully synchronized full-sized instant, third-mirror break-off or linked break-off snapshot (which do not require a cache object). You cannot use a space-optimized instant snapshot.

Creation of space-optimized snapshots that use a shared cache fails if the region size specified for the volume is smaller than the region size set on the cache.

If the region size of a space-optimized snapshot differs from the region size of the cache, this can degrade the system's performance compared to the case where the region sizes are the same.

See “[Creating a shared cache object](#)” on page 215.

The attributes for a snapshot are specified as a tuple to the `vxsnap make` command. This command accepts multiple tuples. One tuple is required for each snapshot that is being created. Each element of a tuple is separated from the next by a slash character (/). Tuples are separated by white space.

To create and manage a space-optimized instant snapshot

- 1 Use the `vxsnap make` command to create a space-optimized instant snapshot. This snapshot can be created by using an existing cache object in the disk group, or a new cache object can be created.
 - To create a space-optimized instant snapshot, `snapvol`, that uses a named shared cache object:

```
# vxsnap [-g diskgroup] make source=vol/newvol=snapvol\  
/cache=cacheobject [alloc=storage_attributes]
```

For example, to create the space-optimized instant snapshot, `snap3myvol`, of the volume, `myvol`, in the disk group, `mydg`, on the disk `mydg14`, and which uses the shared cache object, `cobjmydg`, use the following command:

```
# vxsnap -g mydg make source=myvol/newvol=snap3myvol\  
/cache=cobjmydg alloc=mydg14
```

The DCO is created on the specified allocation.

- To create a space-optimized instant snapshot, `snapvol`, and also create a cache object for it to use:

```
# vxsnap [-g diskgroup] make source=vol/newvol=snapvol\  
[/cachesize=size] [/autogrow=yes] [/ncachemirror=number]\  
[alloc=storage_attributes]
```

The `cachesize` attribute determines the size of the cache relative to the size of the volume. The `autogrow` attribute determines whether VxVM will automatically enlarge the cache if it is in danger of overflowing. By default, `autogrow=on` and the cache is automatically grown.

If `autogrow` is enabled, but the cache cannot be grown, VxVM disables the oldest and largest snapshot that is using the same cache, and releases its cache space for use.

The `ncachemirror` attribute specifies the number of mirrors to create in the cache volume. For backup purposes, the default value of 1 should be sufficient.

For example, to create the space-optimized instant snapshot, `snap4myvol`, of the volume, `myvol`, in the disk group, `mydg`, on the disk `mydg15`, and which uses a newly allocated cache object that is 1GB in size, but which can automatically grow in size, use the following command:

```
# vxsnap -g mydg make source=myvol/new=snap4myvol\  
/cachesize=1g/autogrow=yes alloc=mydg15
```

If a cache is created implicitly by specifying `cachesize`, and `ncachemirror` is specified to be greater than 1, a DCO is attached to the cache volume to enable dirty region logging (DRL). DRL allows fast recovery of the cache backing store after a system crash. The DCO is allocated on the same disks as those that are occupied by the DCO of the source volume. This is done to allow the cache and the source volume to remain in the same disk group for disk group move, split and join operations.

- 2 Use `fsck` (or some utility appropriate for the application running on the volume) to clean the temporary volume's contents. For example, you can use this command with a VxFS file system:

```
# fsck -V vxfs /dev/vx/dsk/diskgroup/snapshot
```

The specified device must have a valid entry in the `/etc/filesystems` file.

- 3 To backup the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape, or to some other backup medium.
- 4 You now have the following options:
 - Refresh the contents of the snapshot. This creates a new point-in-time image of the original volume ready for another backup. If synchronization was already in progress on the snapshot, this operation may result in large portions of the snapshot having to be resynchronized. See [“Refreshing an instant snapshot”](#) on page 233.
 - Restore the contents of the original volume from the snapshot volume. The space-optimized instant snapshot remains intact at the end of the operation. See [“Restoring a volume from an instant snapshot”](#) on page 235.
 - Destroy the snapshot. See [“Removing an instant snapshot”](#) on page 236.

Creating and managing full-sized instant snapshots

Full-sized instant snapshots are not suitable for write-intensive volumes (such as for database redo logs) because the copy-on-write mechanism may degrade the performance of the volume.

For full-sized instant snapshots, you must prepare a volume that is to be used as the snapshot volume. This must be the same size as the volume for which the snapshot is being created, and it must also have the same region size.

See [“Creating a volume for use as a full-sized instant or linked break-off snapshot”](#) on page 217.

The attributes for a snapshot are specified as a tuple to the `vxsnap make` command. This command accepts multiple tuples. One tuple is required for each snapshot that is being created. Each element of a tuple is separated from the next by a slash character (/). Tuples are separated by white space.

To create and manage a full-sized instant snapshot

- 1 To create a full-sized instant snapshot, use the following form of the `vxsnap make` command:

```
# vxsnap [-g diskgroup] make source=volume/snapvol=snapvol\  
[/snapdg=snapdiskgroup] [/syncing=off]
```

The command specifies the volume, *snapvol*, that you prepared earlier.

For example, to use the prepared volume, `snap1myvol`, as the snapshot for the volume, `myvol`, in the disk group, `mydg`, use the following command:

```
# vxsnap -g mydg make source=myvol/snapvol=snap1myvol
```

For full-sized instant snapshots that are created from an empty volume, background synchronization is enabled by default (equivalent to specifying the `syncing=on` attribute). To move a snapshot into a separate disk group, or to turn it into an independent volume, you must wait for its contents to be synchronized with those of its parent volume.

You can use the `vxsnap syncwait` command to wait for the synchronization of the snapshot volume to be completed, as shown here:

```
# vxsnap [-g diskgroup] syncwait snapvol
```

For example, you would use the following command to wait for synchronization to finish on the snapshot volume, `snap2myvol`:

```
# vxsnap -g mydg syncwait snap2myvol
```

This command exits (with a return code of zero) when synchronization of the snapshot volume is complete. The snapshot volume may then be moved to another disk group or turned into an independent volume.

See [“Controlling instant snapshot synchronization”](#) on page 239.

If required, you can use the following command to test if the synchronization of a volume is complete:

```
# vxprint [-g diskgroup] -F%incomplete snapvol
```

This command returns the value `off` if synchronization of the volume, *snapvol*, is complete; otherwise, it returns the value `on`.

You can also use the `vxsnap print` command to check on the progress of synchronization.

See [“Displaying snapshot information”](#) on page 252.

If you do not want to move the snapshot into a separate disk group, or to turn it into an independent volume, specify the `syncing=off` attribute. This avoids unnecessary system overhead.

For example, to turn off synchronization when creating the snapshot of the volume, *myvol*, you would use the following form of the `vxsnap make` command:

```
# vxsnap -g mydg make source=myvol/snapvol=snap1myvol\  
/syncing=off
```

- 2 Use `fsck` (or some utility appropriate for the application running on the volume) to clean the temporary volume’s contents.

For example, you can use this command with a VxFS file system:

```
# fsck -V vxfs /dev/vx/dsk/diskgroup/snapshot
```

The specified device must have a valid entry in the `/etc/filesystems` file.

- 3 To backup the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape, or to some other backup medium.
- 4 You now have the following options:
 - Refresh the contents of the snapshot. This creates a new point-in-time image of the original volume ready for another backup. If synchronization was already in progress on the snapshot, this operation may result in large portions of the snapshot having to be resynchronized.
See [“Refreshing an instant snapshot”](#) on page 233.

- Reattach some or all of the plexes of the snapshot volume with the original volume.
See [“Reattaching an instant snapshot”](#) on page 233.
- Restore the contents of the original volume from the snapshot volume. You can choose whether none, a subset, or all of the plexes of the snapshot volume are returned to the original volume as a result of the operation.
See [“Restoring a volume from an instant snapshot”](#) on page 235.
- Dissociate the snapshot volume entirely from the original volume. This may be useful if you want to use the copy for other purposes such as testing or report generation. If desired, you can delete the dissociated volume.
See [“Dissociating an instant snapshot”](#) on page 236.
- If the snapshot is part of a snapshot hierarchy, you can also choose to split this hierarchy from its parent volumes.
See [“Splitting an instant snapshot hierarchy”](#) on page 237.

Creating and managing third-mirror break-off snapshots

Break-off snapshots are suitable for write-intensive volumes, such as database redo logs.

To turn one or more existing plexes in a volume into a break-off instant snapshot volume, the volume must be a non-layered volume with a `mirror` or `mirror-stripe` layout, or a RAID-5 volume that you have converted to a special layered volume and then mirrored. The plexes in a volume with a `stripe-mirror` layout are mirrored at the subvolume level, and cannot be broken off.

The attributes for a snapshot are specified as a tuple to the `vxsnap make` command. This command accepts multiple tuples. One tuple is required for each snapshot that is being created. Each element of a tuple is separated from the next by a slash character (/). Tuples are separated by white space.

To create and manage a third-mirror break-off snapshot

- 1 To create the snapshot, you can either take some of the existing `ACTIVE` plexes in the volume, or you can use the following command to add new snapshot mirrors to the volume:

```
# vxsnap [-b] [-g diskgroup] addmir volume [nmirror=N] \  
    [alloc=storage_attributes]
```

By default, the `vxsnap addmir` command adds one snapshot mirror to a volume unless you use the `nmirror` attribute to specify a different number of mirrors. The mirrors remain in the `SNAPATT` state until they are fully synchronized. The `-b` option can be used to perform the synchronization in the background. Once synchronized, the mirrors are placed in the `SNAPDONE` state.

For example, the following command adds 2 mirrors to the volume, `vol1`, on disks `mydg10` and `mydg11`:

```
# vxsnap -g mydg addmir vol1 nmirror=2 alloc=mydg10,mydg11
```

If you specify the `-b` option to the `vxsnap addmir` command, you can use the `vxsnap snapwait` command to wait for synchronization of the snapshot plexes to complete, as shown in this example:

```
# vxsnap -g mydg snapwait vol1 nmirror=2
```

- 2 To create a third-mirror break-off snapshot, use the following form of the `vxsnap make` command.

```
# vxsnap [-g diskgroup] make source=volume[/newvol=snapvol]\
{/plex=plex1[,plex2,...]||nmirror=number}
```

Either of the following attributes may be specified to create the new snapshot volume, *snapvol*, by breaking off one or more existing plexes in the original volume:

<code>plex</code>	Specifies the plexes in the existing volume that are to be broken off.
<code>nmirror</code>	Specifies how many plexes are to be broken off. This attribute can only be used with plexes that are in the <code>SNAPDONE</code> state. (Such plexes could have been added to the volume by using the <code>vxsnap addmir</code> command.)

Snapshots that are created from one or more `ACTIVE` or `SNAPDONE` plexes in the volume are already synchronized by definition.

For backup purposes, a snapshot volume with one plex should be sufficient.

For example, to create the instant snapshot volume, *snap2myvol*, of the volume, *myvol*, in the disk group, *mydg*, from a single existing plex in the volume, use the following command:

```
# vxsnap -g mydg make source=myvol/newvol=snap2myvol/nmirror=1
```

The next example shows how to create a mirrored snapshot from two existing plexes in the volume:

```
# vxsnap -g mydg make source=myvol/newvol=snap2myvol/plex=myvol-03,myvol-04
```

- 3 Use `fsck` (or some utility appropriate for the application running on the volume) to clean the temporary volume's contents. For example, you can use this command with a VxFS file system:

```
# fsck -V vxfs /dev/vx/dsk/diskgroup/snapshot
```

The specified device must have a valid entry in the `/etc/filesystems` file.

- 4 To backup the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape, or to some other backup medium.
- 5 You now have the following options:

- Refresh the contents of the snapshot. This creates a new point-in-time image of the original volume ready for another backup. If synchronization was already in progress on the snapshot, this operation may result in large portions of the snapshot having to be resynchronized. See [“Refreshing an instant snapshot”](#) on page 233.
- Reattach some or all of the plexes of the snapshot volume with the original volume. See [“Reattaching an instant snapshot”](#) on page 233.
- Restore the contents of the original volume from the snapshot volume. You can choose whether none, a subset, or all of the plexes of the snapshot volume are returned to the original volume as a result of the operation. See [“Restoring a volume from an instant snapshot”](#) on page 235.
- Dissociate the snapshot volume entirely from the original volume. This may be useful if you want to use the copy for other purposes such as testing or report generation. If desired, you can delete the dissociated volume. See [“Dissociating an instant snapshot”](#) on page 236.
- If the snapshot is part of a snapshot hierarchy, you can also choose to split this hierarchy from its parent volumes. See [“Splitting an instant snapshot hierarchy”](#) on page 237.

Creating and managing linked break-off snapshot volumes

Linked break-off snapshots are suitable for write-intensive volumes. Specifically, they are used for off-host processing, because the snapshot could be in a different disk group to start with and could avoid disk group split/join operations

For linked break-off snapshots, you must prepare a volume that is to be used as the snapshot volume. This must be the same size as the volume for which the snapshot is being created, and it must also have the same region size.

See [“Creating a volume for use as a full-sized instant or linked break-off snapshot”](#) on page 217.

The attributes for a snapshot are specified as a tuple to the `vxsnap make` command. This command accepts multiple tuples. One tuple is required for each snapshot that is being created. Each element of a tuple is separated from the next by a slash character (/). Tuples are separated by white space.

To create and manage a linked break-off snapshot

- 1 Use the following command to link the prepared snapshot volume, *snapvol*, to the data volume:

```
# vxsnap [-g diskgroup] [-b] addmir volume mirvol=snapvol \  
[mirdg=snapdg]
```

The optional *mirdg* attribute can be used to specify the snapshot volume's current disk group, *snapdg*. The *-b* option can be used to perform the synchronization in the background. If the *-b* option is not specified, the command does not return until the link becomes *ACTIVE*.

For example, the following command links the prepared volume, *prepsnap*, in the disk group, *mysnapdg*, to the volume, *vol1*, in the disk group, *mydg*:

```
# vxsnap -g mydg -b addmir vol1 mirvol=prepsnap mirdg=mysnapdg
```

If the *-b* option is specified, you can use the *vxsnap snapwait* command to wait for the synchronization of the linked snapshot volume to complete, as shown in this example:

```
# vxsnap -g mydg snapwait vol1 mirvol=prepsnap mirdg=mysnapvoldg
```

- 2 To create a linked break-off snapshot, use the following form of the *vxsnap make* command.

```
# vxsnap [-g diskgroup] make source=volume/snapvol=snapvol \  
[/snapdg=snapdiskgroup]
```

The *snapdg* attribute must be used to specify the snapshot volume's disk group if this is different from that of the data volume.

For example, to use the prepared volume, *prepsnap*, as the snapshot for the volume, *vol1*, in the disk group, *mydg*, use the following command:

```
# vxsnap -g mydg make source=vol1/snapvol=prepsnap/snapdg=mysnapdg
```

- 3 Use *fsck* (or some utility appropriate for the application running on the volume) to clean the temporary volume's contents. For example, you can use this command with a VxFS file system:

```
# fsck -V vxfs /dev/vx/dsk/diskgroup/snapshot
```

The specified device must have a valid entry in the */etc/filesystems* file.

- 4 To backup the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape, or to some other backup medium.
- 5 You now have the following options:
 - Refresh the contents of the snapshot. This creates a new point-in-time image of the original volume ready for another backup. If synchronization was already in progress on the snapshot, this operation may result in large portions of the snapshot having to be resynchronized. See [“Refreshing an instant snapshot”](#) on page 233.
 - Reattach the snapshot volume with the original volume. See [“Reattaching a linked break-off snapshot volume”](#) on page 234.
 - Dissociate the snapshot volume entirely from the original volume. This may be useful if you want to use the copy for other purposes such as testing or report generation. If desired, you can delete the dissociated volume. See [“Dissociating an instant snapshot”](#) on page 236.
 - If the snapshot is part of a snapshot hierarchy, you can also choose to split this hierarchy from its parent volumes. See [“Splitting an instant snapshot hierarchy”](#) on page 237.

Creating multiple instant snapshots

You can create multiple instant snapshots for all volumes that form a consistent group. The `vxsnap make` command accepts multiple tuples that define the source and snapshot volumes names as their arguments.

For example, to create three instant snapshots, each with the same redundancy, from specified storage, the following form of the command can be used:

```
# vxsnap [-g diskgroup] make source=vol1/snapvol=snapvol1\  
source=vol2/snapvol=snapvol2 source=vol3/snapvol=snapvol3
```

The snapshot volumes (*snapvol1*, *snapvol2* and so on) must have been prepared in advance.

See [“Creating a volume for use as a full-sized instant or linked break-off snapshot”](#) on page 217.

The specified source volumes (*vol1*, *vol2* and so on) may be the same volume or they can be different volumes.

If all the snapshots are to be space-optimized and to share the same cache, the following form of the command can be used:

```
# vxsnap [-g diskgroup] make \
  source=vol1/newvol=snapvol1/cache=cacheobj \
  source=vol2/newvol=snapvol2/cache=cacheobj \
  source=vol3/newvol=snapvol3/cache=cacheobj \
  [alloc=storage_attributes]
```

The `vxsnap make` command also allows the snapshots to be of different types, have different redundancy, and be configured from different storage, as shown here:

```
# vxsnap [-g diskgroup] make source=vol1/snapvol=snapvol1 \
  source=vol2[/newvol=snapvol2]/cache=cacheobj\
  [/alloc=storage_attributes2] [/nmirror=number2]
  source=vol3[/newvol=snapvol3] [/alloc=storage_attributes3]\
  /nmirror=number3
```

In this example, *snapvol1* is a full-sized snapshot that uses a prepared volume, *snapvol2* is a space-optimized snapshot that uses a prepared cache, and *snapvol3* is a break-off full-sized snapshot that is formed from plexes of the original volume.

An example of where you might want to create mixed types of snapshots at the same time is when taking snapshots of volumes containing database redo logs and database tables:

```
# vxsnap -g mydg make \
  source=logv1/newvol=snplogv1/drl=sequential/nmirror=1 \
  source=logv2/newvol=snplogv2/drl=sequential/nmirror=1 \
  source=datav1/newvol=snpdatav1/cache=mydgcobj/drl=on \
  source=datav2/newvol=snpdatav2/cache=mydgcobj/drl=on
```

In this example, sequential DRL is enabled for the snapshots of the redo log volumes, and normal DRL is applied to the snapshots of the volumes that contain the database tables. The two space-optimized snapshots are configured to share the same cache object in the disk group. Also note that break-off snapshots are used for the redo logs as such volumes are write intensive.

Creating instant snapshots of volume sets

Volume set names can be used in place of volume names with the following `vxsnap` operations on instant snapshots: `addmir`, `dis`, `make`, `prepare`, `reattach`, `refresh`, `restore`, `rmmir`, `split`, `syncpause`, `syncresume`, `syncstart`, `syncstop`, `syncwait`, and `unprepare`.

The procedure for creating an instant snapshot of a volume set is the same as that for a standalone volume. However, there are certain restrictions if a full-sized

instant snapshot is to be created from a prepared volume set. A full-sized instant snapshot of a volume set must itself be a volume set with the same number of volumes, and the same volume sizes and index numbers as the parent.

For example, if a volume set contains three volumes with sizes 1GB, 2GB and 3GB, and indexes 0, 1 and 2 respectively, then the snapshot volume set must have three volumes with the same sizes matched to the same set of index numbers. The corresponding volumes in the parent and snapshot volume sets are also subject to the same restrictions as apply between standalone volumes and their snapshots.

You can use the `vxvset list` command to verify that the volume sets have identical characteristics as shown in this example:

```
# vxvset -g mydg list vset1
```

VOLUME	INDEX	LENGTH	KSTATE	CONTEXT
vol_0	0	204800	ENABLED	-
vol_1	1	409600	ENABLED	-
vol_2	2	614400	ENABLED	-

```
# vxvset -g mydg list snapvset1
```

VOLUME	INDEX	LENGTH	KSTATE	CONTEXT
svol_0	0	204800	ENABLED	-
svol_1	1	409600	ENABLED	-
svol_2	2	614400	ENABLED	-

A full-sized instant snapshot of a volume set can be created using a prepared volume set in which each volume is the same size as the corresponding volume in the parent volume set. Alternatively, you can use the `nmirrors` attribute to specify the number of plexes that are to be broken off provided that sufficient plexes exist for each volume in the volume set.

The following example shows how to prepare a source volume set, `vset1`, and an identical volume set, `snapvset1`, which is then used to create the snapshot:

```
# vxsnap -g mydg prepare vset1
# vxsnap -g mydg prepare snapvset1
# vxsnap -g mydg make source=vset1/snapvol=snapvset1
```

To create a full-sized third-mirror break-off snapshot, you must ensure that each volume in the source volume set contains sufficient plexes. The following example shows how to achieve this by using the `vxsnap` command to add the required number of plexes before breaking off the snapshot:

```
# vxsnap -g mydg prepare vset2
# vxsnap -g mydg addmir vset2 nmirror=1
# vxsnap -g mydg make source=vset2/newvol=snapvset2/nmirror=1
```

See [“Adding snapshot mirrors to a volume”](#) on page 231.

To create a space-optimized instant snapshot of a volume set, the commands are again identical to those for a standalone volume as shown in these examples:

```
# vxsnap -g mydg prepare vset3
# vxsnap -g mydg make source=vset3/newvol=snapvset3/cachesize=20m

# vxsnap -g mydg prepare vset4
# vxsnap -g mydg make source=vset4/newvol=snapvset4/cache=mycobj
```

Here a new cache object is created for the volume set, `vset3`, and an existing cache object, `mycobj`, is used for `vset4`.

Adding snapshot mirrors to a volume

If you are going to create a full-sized break-off snapshot volume, you can use the following command to add new snapshot mirrors to a volume:

```
# vxsnap [-b] [-g diskgroup] addmir volume|volume_set \
    [nmirror=N] [alloc=storage_attributes]
```

The volume must have been prepared using the `vxsnap prepare` command.

If a volume set name is specified instead of a volume, the specified number of plexes is added to each volume in the volume set.

By default, the `vxsnap addmir` command adds one snapshot mirror to a volume unless you use the `nmirror` attribute to specify a different number of mirrors. The mirrors remain in the `SNAPATT` state until they are fully synchronized. The `-b` option can be used to perform the synchronization in the background. Once synchronized, the mirrors are placed in the `SNAPDONE` state.

For example, the following command adds 2 mirrors to the volume, `vol1`, on disks `mydg10` and `mydg11`:

```
# vxsnap -g mydg addmir vol1 nmirror=2 alloc=mydg10,mydg11
```

This command is similar in usage to the `vxassist snapstart` command, and supports the traditional third-mirror break-off snapshot model. As such, it does not provide an instant snapshot capability.

Once you have added one or more snapshot mirrors to a volume, you can use the `vxsnap make` command with either the `nmirror` attribute or the `plex` attribute to create the snapshot volumes.

Removing a snapshot mirror

To remove a single snapshot mirror from a volume, use this command:

```
# vxsnap [-g diskgroup] rmmir volume|volume_set
```

For example, the following command removes a snapshot mirror from the volume, `voll`:

```
# vxsnap -g mydg rmmir voll
```

This command is similar in usage to the `vxassist snapabort` command.

If a volume set name is specified instead of a volume, a mirror is removed from each volume in the volume set.

Removing a linked break-off snapshot volume

To remove a linked break-off snapshot volume from a volume, use this command:

```
# vxsnap [-g diskgroup] rmmir volume|volume_set mirvol=snapvol \  
[mir dg=snappediskgroup]
```

The `mirvol` and optional `mir dg` attributes specify the snapshot volume, *snapvol*, and its disk group, *snappediskgroup*. For example, the following command removes a linked snapshot volume, `prepsnap`, from the volume, `voll`:

```
# vxsnap -g mydg rmmir voll mirvol=prepsnap mir dg=mynapdg
```

Adding a snapshot to a cascaded snapshot hierarchy

To create a snapshot and push it onto a snapshot hierarchy between the original volume and an existing snapshot volume, specify the name of the existing snapshot volume as the value of the `infrontof` attribute to the `vxsnap make` command.

The following example shows how to place the space-optimized snapshot, `thurs_bu`, of the volume, `dbvol`, in front of the earlier snapshot, `wed_bu`:

```
# vxsnap -g dbdg make source=dbvol/newvol=thurs_bu/  
infrontof=wed_bu/cache=dbdgcache
```

Similarly, the next snapshot that is taken, `fri_bu`, is placed in front of `thurs_bu`:

```
# vxsnap -g dbdg make source=dbvol/newvol=fri_bu/\
infrontof=thurs_bu/cache=dbdgcache
```

See “[Controlling instant snapshot synchronization](#)” on page 239.

Refreshing an instant snapshot

Refreshing an instant snapshot replaces it with another point-in-time copy of a parent volume. To refresh one or more snapshots and make them immediately available for use, use the following command:

```
# vxsnap [-g diskgroup] refresh snapvolume|snapvolume_set \
[source=volume|volume_set] [snapvol2 [source=vol2]...] \
[syncing=yes|no]
```

If the source volume is not specified, the immediate parent of the snapshot is used. For full-sized instant snapshots, resynchronization is started by default. To disable resynchronization, specify the `syncing=no` attribute. This attribute is not supported for space-optimized snapshots.

Warning: The snapshot that is being refreshed must not be open to any application. For example, any file system configured on the volume must first be unmounted.

It is possible to refresh a volume from an unrelated volume provided that their sizes are compatible.

You can use the `vxsnap syncwait` command to wait for the synchronization of the snapshot volume to be completed, as shown here:

```
# vxsnap [-g diskgroup] syncwait snapvol
```

See “[Controlling instant snapshot synchronization](#)” on page 239.

Reattaching an instant snapshot

Note: This operation is not supported for space-optimized instant snapshots.

Using the following command, some or all plexes of an instant snapshot may be reattached to the specified original volume, or to a source volume in the snapshot hierarchy above the snapshot volume:

```
# vxsnap [-g diskgroup] reattach snapvolume|snapvolume_set \
source=volume|volume_set [nmirror=number]
```

By default, all the plexes are reattached, which results in the removal of the snapshot. If required, the number of plexes to be reattached may be specified as the value assigned to the `nmirror` attribute.

Warning: The snapshot that is being reattached must not be open to any application. For example, any file system configured on the snapshot volume must first be unmounted.

It is possible to reattach a volume to an unrelated volume provided that their volume sizes and region sizes are compatible.

For example the following command reattaches one plex from the snapshot volume, `snapmyvol`, to the volume, `myvol`:

```
# vxsnap -g mydg reattach snapmyvol source=myvol nmirror=1
```

While the reattached plexes are being resynchronized from the data in the parent volume, they remain in the `SNAPTMP` state. After resynchronization is complete, the plexes are placed in the `SNAPDONE` state. You can use the `vxsnap snapwait` command (but not `vxsnap syncwait`) to wait for the resynchronization of the reattached plexes to complete, as shown here:

```
# vxsnap -g mydg snapwait myvol nmirror=1
```

If the volume and its snapshot have both been resized (to an identical smaller or larger size) before performing the reattachment, a fast resynchronization can still be performed. A full resynchronization is not required. Version 20 DCO volumes are resized proportionately when the associated data volume is resized. For version 0 DCO volumes, the FastResync maps stay the same size, but the region size is recalculated, and the locations of the dirty bits in the existing maps are adjusted. In both cases, new regions are marked as dirty in the maps.

Reattaching a linked break-off snapshot volume

Unlike other types of snapshot, the reattachment operation for linked break-off snapshot volumes does not return the plexes of the snapshot volume to the parent volume. The link relationship is re-established that makes the snapshot volume a mirror of the parent volume, and this allows the snapshot data to be resynchronized.

To reattach a linked break-off snapshot volume, use the following form of the `vxsnap reattach` command:

```
# vxsnap [-g snapdiskgroup] reattach snapvolume|snapvolume_set \  
source=volume|volume_set [sourcedg=diskgroup]
```

The `sourcedg` attribute must be used to specify the data volume's disk group if this is different from the snapshot volume's disk group, *snapdiskgroup*.

Warning: The snapshot that is being reattached must not be open to any application. For example, any file system configured on the snapshot volume must first be unmounted.

It is possible to reattach a volume to an unrelated volume provided that their sizes and region sizes are compatible.

For example the following command reattaches the snapshot volume, `prepsnap`, in the disk group, `snapdg`, to the volume, `myvol`, in the disk group, `mydg`:

```
# vxsnap -g snapdg reattach prepsnap source=myvol sourcedg=mydg
```

After resynchronization of the snapshot volume is complete, the link is placed in the `ACTIVE` state. You can use the `vxsnap snapwait` command (but not `vxsnap syncwait`) to wait for the resynchronization of the reattached volume to complete, as shown here:

```
# vxsnap -g snapdg snapwait myvol mirvol=prepsnap
```

Restoring a volume from an instant snapshot

It may sometimes be desirable to reinstate the contents of a volume from a backup or modified replica in a snapshot volume. The following command may be used to restore one or more volumes from the specified snapshots:

```
# vxsnap [-g diskgroup] restore volume|volume_set \
  source=snapvolume|snapvolume_set \
  [[volume2|volume_set2 \
  source=snapvolume2|snapvolume_set2]...] \
  [destroy=yes|no] [syncing=yes|no] [nmirror=number]
```

For a full-sized instant snapshot, some or all of its plaxes may be reattached to the parent volume or to a specified source volume in the snapshot hierarchy above the snapshot volume. If `destroy=yes` is specified, all the plaxes of the full-sized instant snapshot are reattached and the snapshot volume is removed.

For a space-optimized instant snapshot, the cached data is used to recreate the contents of the specified volume. The space-optimized instant snapshot remains unchanged by the `restore` operation.

Warning: For this operation to succeed, the volume that is being restored and the snapshot volume must not be open to any application. For example, any file systems that are configured on either volume must first be unmounted.

It is not possible to restore a volume from an unrelated volume.

The `destroy` and `nmirror` attributes are not supported for space-optimized instant snapshots.

The following example demonstrates how to restore the volume, `myvol`, from the space-optimized snapshot, `snap3myvol`.

```
# vxsnap -g mydg restore myvol source=snap3myvol
```

Dissociating an instant snapshot

The following command breaks the association between a full-sized instant snapshot volume, `snapvol`, and its parent volume, so that the snapshot may be used as an independent volume:

```
# vxsnap [-f] [-g diskgroup] dis snapvolume|snapvolume_set
```

This operation fails if the snapshot, `snapvol`, has unsynchronized snapshots. If this happens, the dependent snapshots must be fully synchronized from `snapvol`. When no dependent snapshots remain, `snapvol` may be dissociated. The snapshot hierarchy is then adopted by the parent volume of `snapvol`.

See [“Controlling instant snapshot synchronization”](#) on page 239.

See [“Removing an instant snapshot”](#) on page 236.

The following command dissociates the snapshot, `snap2myvol`, from its parent volume:

```
# vxsnap -g mydg dis snap2myvol
```

Warning: When applied to a volume set or to a component volume of a volume set, this operation can result in inconsistencies in the snapshot hierarchy in the case of a system crash or hardware failure. If the operation is applied to a volume set, the `-f` (force) option must be specified.

Removing an instant snapshot

When you have dissociated a full-sized instant snapshot, you can use the `vxedit` command to delete it altogether, as shown in this example:

```
# vxedit -g mydg -r rm snap2myvol
```

You can also use this command to remove a space-optimized instant snapshot from its cache.

See [“Removing a cache”](#) on page 244.

Splitting an instant snapshot hierarchy

Note: This operation is not supported for space-optimized instant snapshots.

The following command breaks the association between a snapshot hierarchy that has the snapshot volume, *snapvol*, at its head, and its parent volume, so that the snapshot hierarchy may be used independently of the parent volume:

```
# vxsnap [-f] [-g diskgroup] split snapvolume|snapvolume_set
```

The topmost snapshot volume in the hierarchy must have been fully synchronized for this command to succeed. Snapshots that are lower down in the hierarchy need not have been fully resynchronized.

See [“Controlling instant snapshot synchronization”](#) on page 239.

The following command splits the snapshot hierarchy under *snap2myvol* from its parent volume:

```
# vxsnap -g mydg split snap2myvol
```

Warning: When applied to a volume set or to a component volume of a volume set, this operation can result in inconsistencies in the snapshot hierarchy in the case of a system crash or hardware failure. If the operation is applied to a volume set, the *-f* (force) option must be specified.

Displaying instant snapshot information

The `vxsnap print` command may be used to display information about the snapshots that are associated with a volume.

```
# vxsnap [-g diskgroup] print [vol]
```

This command shows the percentage progress of the synchronization of a snapshot or volume. If no volume is specified, information about the snapshots for all the volumes in a disk group is displayed. The following example shows a volume,

vol1, which has a full-sized snapshot, snapvol1 whose contents have not been synchronized with vol1:

```
# vxsnap -g mydg print
NAME          SNAPOBJECT      TYPE    PARENT  SNAPSHOT  %DIRTY  %VALID
vol1          --              volume  --      --        --      100
              snapvol1_snp1  volume  --      snapvol1  1.30    --
snapvol1     vol1_snp1      volume  vol1    --        1.30    1.30
```

The %DIRTY value for snapvol1 shows that its contents have changed by 1.30% when compared with the contents of vol1. As snapvol1 has not been synchronized with vol1, the %VALID value is the same as the %DIRTY value. If the snapshot were partly synchronized, the %VALID value would lie between the %DIRTY value and 100%. If the snapshot were fully synchronized, the %VALID value would be 100%. The snapshot could then be made independent or moved into another disk group.

Additional information about the snapshots of volumes and volume sets can be obtained by using the -n option with the vxsnap print command:

```
# vxsnap [-g diskgroup] -n [-l] [-v] [-x] print [vol]
```

Alternatively, you can use the vxsnap list command, which is an alias for the vxsnap -n print command:

```
# vxsnap [-g diskgroup] [-l] [-v] [-x] list [vol]
```

The following output is an example of using this command on the disk group dg1:

```
# vxsnap -g dg -vx list
NAME  DG    OBJTYPE  SNAPTYPE  PARENT  PARENTDG  SNAPDATE          CHANGE_DATA  SYNCED_DATA
vol   dg1   vol      -          -        -          -                -            10G (100%)
svol1 dg2   vol      fullinst  vol     dg1        2006/2/1 12:29  20M (0.2%)  60M (0.6%)
svol2 dg1   vol      mirbrk    vol     dg1        2006/2/1 12:29  120M (1.2%) 10G (100%)
svol3 dg2   vol      volbrk    vol     dg1        2006/2/1 12:29  105M (1.1%) 10G (100%)
svol21 dg1   vol      spaceopt  svol2   dg1        2006/2/1 12:29  52M (0.5%)  52M (0.5%)
vol-02 dg1   plex     snapmir   vol     dg1        -              -            56M (0.6%)
mvol  dg2   vol      mirvol    vol     dg1        -              -            58M (0.6%)
vset1 dg1   vset     -         -        -          -                -            2G (100%)
v1    dg1   compvol  -         -        -          -                -            1G (100%)
v2    dg1   compvol  -         -        -          -                -            1G (100%)
svset1 dg1   vset     mirbrk    vset    dg1        2006/2/1 12:29  1G (50%)    2G (100%)
sv1   dg1   compvol  mirbrk    v1      dg1        2006/2/1 12:29  512M (50%) 1G (100%)
sv2   dg1   compvol  mirbrk    v2      dg1        2006/2/1 12:29  512M (50%) 1G (100%)
```

vol-03	dg1	plex	detmir	vol	dg1	-	20M (0.2%)	-
mvol2	dg2	vol	detvol	vol	dg1	-	20M (0.2%)	-

This shows that the volume `vol` has three full-sized snapshots, `svol1`, `svol2` and `svol3`, which are of types full-sized instant (`fullinst`), mirror break-off (`mirbrk`) and linked break-off (`volbrk`). It also has one snapshot `plex` (`snapmir`), `vol-02`, and one linked mirror volume (`mirvol`), `mvol`. The snapshot `svol2` itself has a space-optimized instant snapshot (`spaceopt`), `svol21`. There is also a volume set, `vset1`, with component volumes `v1` and `v2`. This volume set has a mirror break-off snapshot, `svset1`, with component volumes `sv1` and `sv2`. The last two entries show a detached plex, `vol-03`, and a detached mirror volume, `mvol2`, which have `vol` as their parent volume. These snapshot objects may have become detached due to an I/O error, or, in the case of the plex, by running the `vxplex det` command.

The `CHANGE_DATA` column shows the approximate difference between the current contents of the snapshot and its parent volume. This corresponds to the amount of data that would have to be resynchronized to make the contents the same again.

The `SYNCED_DATA` column shows the approximate progress of synchronization since the snapshot was taken.

The `-l` option can be used to obtain a longer form of the output listing instead of the tabular form.

The `-x` option expands the output to include the component volumes of volume sets.

See the `vxsnap(1M)` manual page for more information about using the `vxsnap print` and `vxsnap list` commands.

Controlling instant snapshot synchronization

Synchronization of the contents of a snapshot with its original volume is not possible for space-optimized instant snapshots.

By default, synchronization is enabled for the `vxsnap reattach`, `refresh` and `restore` operations on instant snapshots. Otherwise, synchronization is disabled unless you specify the `syncing=yes` attribute to the `vxsnap` command.

[Table 17-1](#) shows the commands that are provided for controlling the synchronization manually.

Table 17-1 Commands for controlling instant snapshot synchronization

Command	Description
<code>vxsnap [-g <i>diskgroup</i>] syncpause \</code> <code><i>vol</i> <i>vol_set</i></code>	Pause synchronization of a volume.
<code>vxsnap [-g <i>diskgroup</i>] syncresume \</code> <code><i>vol</i> <i>vol_set</i></code>	Resume synchronization of a volume.
<code>vxsnap [-b] [-g <i>diskgroup</i>] syncstart \</code> <code><i>vol</i> <i>vol_set</i></code>	Start synchronization of a volume. The <code>-b</code> option puts the operation in the background.
<code>vxsnap [-g <i>diskgroup</i>] syncstop \</code> <code><i>vol</i> <i>vol_set</i></code>	Stop synchronization of a volume.
<code>vxsnap [-g <i>diskgroup</i>] syncwait \</code> <code><i>vol</i> <i>vol_set</i></code>	Exit when synchronization of a volume is complete. An error is returned if the <code>vol</code> or <code>vol_set</code> is invalid (for example, it is a space-optimized snapshot), or if the <code>vol</code> or <code>vol_set</code> is not being synchronized. Note: You cannot use this command to wait for synchronization of reattached plexes to complete.

The commands that are shown in [Table 17-1](#) cannot be used to control the synchronization of linked break-off snapshots.

The `vxsnap snapwait` command is provided to wait for the link between new linked break-off snapshots to become ACTIVE, or for reattached snapshot plexes to reach the SNAPDONE state following resynchronization.

See [“Creating and managing linked break-off snapshot volumes”](#) on page 226.

See [“Reattaching an instant snapshot”](#) on page 233.

See [“Reattaching a linked break-off snapshot volume”](#) on page 234.

Improving the performance of snapshot synchronization

The following optional arguments to the `-o` option are provided to help optimize the performance of synchronization when using the `make`, `refresh`, `restore` and `syncstart` operations with full-sized instant snapshots:

<code>iosize=size</code>	Specifies the size of each I/O request that is used when synchronizing the regions of a volume. Specifying a larger size causes synchronization to complete sooner, but with greater impact on the performance of other processes that are accessing the volume. The default size of 1m (1MB) is suggested as the minimum value for high-performance array and controller hardware. The specified value is rounded to a multiple of the volume's region size.
<code>slow=iodelay</code>	Specifies the delay in milliseconds between synchronizing successive sets of regions as specified by the value of <code>iosize</code> . This can be used to change the impact of synchronization on system performance. The default value of <code>iodelay</code> is 0 milliseconds (no delay). Increasing this value slows down synchronization, and reduces the competition for I/O bandwidth with other processes that may be accessing the volume.

Options may be combined as shown in the following examples:

```
# vxsnap -g mydg -o iosize=2m,slow=100 make \  
  source=myvol/snapvol=snap2myvol/syncing=on  
  
# vxsnap -g mydg -o iosize=10m,slow=250 syncstart snap2myvol
```

Note: The `iosize` and `slow` parameters are not supported for space-optimized snapshots.

Listing the snapshots created on a cache

To list the space-optimized instant snapshots that have been created on a cache object, use the following command:

```
# vxcache [-g diskgroup] listvol cache_object
```

The snapshot names are printed as a space-separated list ordered by timestamp. If two or more snapshots have the same timestamp, these snapshots are sorted in order of decreasing size.

Tuning the autogrow attributes of a cache

The `highwatermark`, `autogrowby` and `maxautogrow` attributes determine how the VxVM cache daemon (`vxcached`) maintains the cache if the `autogrow` feature has been enabled and `vxcached` is running:

- When cache usage reaches the high watermark value, `highwatermark` (default value is 90 percent), `vxcached` grows the size of the cache volume by the value of `autogrowby` (default value is 20% of the size of the cache volume in blocks). The new required cache size cannot exceed the value of `maxautogrow` (default value is twice the size of the cache volume in blocks).
- When cache usage reaches the high watermark value, and the new required cache size would exceed the value of `maxautogrow`, `vxcached` deletes the oldest snapshot in the cache. If there are several snapshots with the same age, the largest of these is deleted.

If the `autogrow` feature has been disabled:

- When cache usage reaches the high watermark value, `vxcached` deletes the oldest snapshot in the cache. If there are several snapshots with the same age, the largest of these is deleted. If there is only a single snapshot, this snapshot is detached and marked as invalid.

Note: The `vxcached` daemon does not remove snapshots that are currently open, and it does not remove the last or only snapshot in the cache.

If the cache space becomes exhausted, the snapshot is detached and marked as invalid. If this happens, the snapshot is unrecoverable and must be removed. Enabling the `autogrow` feature on the cache helps to avoid this situation occurring. However, for very small caches (of the order of a few megabytes), it is possible for the cache to become exhausted before the system has time to respond and grow the cache. In such cases, you can increase the size of the cache manually.

Alternatively, you can use the `vxcache set` command to reduce the value of `highwatermark` as shown in this example:

```
# vxcache -g mydg set highwatermark=60 cobjmydg
```

You can use the `maxautogrow` attribute to limit the maximum size to which a cache can grow. To estimate this size, consider how much the contents of each source volume are likely to change between snapshot refreshes, and allow some additional space for contingency.

If necessary, you can use the `vxcache set` command to change other `autogrow` attribute values for a cache.

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

Monitoring and displaying cache usage

You can use the `vxcache stat` command to display cache usage.

For example, to see how much space is used and how much remains available in all cache objects in the diskgroup `mydg`, enter the following:

```
# vxcache -g mydg stat
```

Growing and shrinking a cache

You can use the `vxcache` command to increase the size of the cache volume that is associated with a cache object:

```
# vxcache [-g diskgroup] growcacheto cache_object
      size
```

For example, to increase the size of the cache volume associated with the cache object, `mycache`, to 2GB, you would use the following command:

```
# vxcache -g mydg growcacheto mycache 2g
```

To grow a cache by a specified amount, use the following form of the command shown here:

```
# vxcache [-g diskgroup] growcacheby cache_object
      size
```

For example, the following command increases the size of `mycache` by 1GB:

```
# vxcache -g mydg growcacheby mycache 1g
```

You can similarly use the `shrinkcacheby` and `shrinkcacheto` operations to reduce the size of a cache.

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

Removing a cache

To remove a cache completely, including the cache object, its cache volume and all space-optimized snapshots that use the cache:

- 1 Run the following command to find out the names of the top-level snapshot volumes that are configured on the cache object:

```
# vxprint -g diskgroup -vne \  
    "v_plex.pl_subdisk.sd_dm_name ~ /cache_object/"
```

where *cache_object* is the name of the cache object.

- 2 Remove all the top-level snapshots and their dependent snapshots (this can be done with a single command):

```
# vxedit -g diskgroup -r rm snapvol ...
```

where *snapvol* is the name of a top-level snapshot volume.

- 3 Stop the cache object:

```
# vxcache -g diskgroup stop cache_object
```

- 4 Finally, remove the cache object and its cache volume:

```
# vxedit -g diskgroup -r rm cache_object
```

Creating traditional third-mirror break-off snapshots

VxVM provides third-mirror break-off snapshot images of volume devices using `vxassist` and other commands.

To enhance the efficiency and usability of volume snapshots, turn on FastResync.

See the *Veritas Volume Manager Administrator's Guide*.

If Persistent FastResync is required, you must associate a version 0 DCO with the volume.

See [“Adding a version 0 DCO and DCO volume”](#) on page 253.

A plex is required that is large enough to store the complete contents of the volume. Alternatively, you can use space-optimized instant snapshots.

The recommended approach to performing volume backup from the command line, or from a script, is to use the `vxsnap` command. The `vxassist snapstart`, `snapshotwait`, and `snapshot` commands are supported for backward compatibility.

The `vxassist snapshot` procedure consists of two steps:

- Run `vxassist snapstart` to create a snapshot mirror.
- Run `vxassist snapshot` to create a snapshot volume.

The `vxassist snapstart` step creates a write-only backup plex which gets attached to and synchronized with the volume. When synchronized with the volume, the backup plex is ready to be used as a `snapshot mirror`. The end of the update procedure is indicated by the new `snapshot mirror` changing its state to `SNAPDONE`. This change can be tracked by the `vxassist snapwait` task, which waits until at least one of the mirrors changes its state to `SNAPDONE`. If the attach process fails, the `snapshot mirror` is removed and its space is released.

Note: If the `snapstart` procedure is interrupted, the snapshot mirror is automatically removed when the volume is started.

Once the `snapshot mirror` is synchronized, it continues being updated until it is detached. You can then select a convenient time at which to create a `snapshot volume` as an image of the existing volume. You can also ask users to refrain from using the system during the brief time required to perform the `snapshot` (typically less than a minute). The amount of time involved in creating the `snapshot mirror` is long in contrast to the brief amount of time that it takes to create the `snapshot volume`.

The online backup procedure is completed by running the `vxassist snapshot` command on a volume with a `SNAPDONE` mirror. This task detaches the finished `snapshot` (which becomes a normal mirror), creates a new normal volume and attaches the `snapshot mirror` to the `snapshot volume`. The `snapshot` then becomes a normal, functioning volume and the state of the `snapshot` is set to `ACTIVE`.

To back up a volume using the `vxassist` command

- 1 Create a snapshot mirror for a volume using the following command:

```
# vxassist [-b] [-g diskgroup] snapstart [nmirror=N] volume
```

For example, to create a snapshot mirror of a volume called `voldef`, use the following command:

```
# vxassist [-g diskgroup] snapstart voldef
```

The `vxassist snapstart` task creates a write-only mirror, which is attached to and synchronized from the volume to be backed up.

By default, VxVM attempts to avoid placing snapshot mirrors on a disk that already holds any plexes of a data volume. However, this may be impossible if insufficient space is available in the disk group. In this case, VxVM uses any available space on other disks in the disk group. If the snapshot plexes are placed on disks which are used to hold the plexes of other volumes, this may cause problems when you subsequently attempt to move a snapshot volume into another disk group.

To override the default storage allocation policy, you can use storage attributes to specify explicitly which disks to use for the snapshot plexes.

If you start `vxassist snapstart` in the background using the `-b` option, you can use the `vxassist snapwait` command to wait for the creation of the mirror to complete as shown here:

```
# vxassist [-g diskgroup] snapwait volume
```

If `vxassist snapstart` is not run in the background, it does not exit until the mirror has been synchronized with the volume. The mirror is then ready to be used as a plex of a snapshot volume. While attached to the original volume, its contents continue to be updated until you take the snapshot.

Use the `nmirror` attribute to create as many snapshot mirrors as you need for the snapshot volume. For a backup, you should usually only require the default of one.

It is also possible to make a snapshot plex from an existing plex in a volume.

See [“Converting a plex into a snapshot plex”](#) on page 248.

- 2 Choose a suitable time to create a snapshot. If possible, plan to take the snapshot at a time when users are accessing the volume as little as possible.

3 Create a snapshot volume using the following command:

```
# vxassist [-g diskgroup] snapshot [nmirror=N] volume snapshot
```

If required, use the `nmirror` attribute to specify the number of mirrors in the snapshot volume.

For example, to create a snapshot of `voldef`, use the following command:

```
# vxassist -g mydg snapshot voldef snapvoldef
```

The `vxassist snapshot` task detaches the finished snapshot mirror, creates a new volume, and attaches the snapshot mirror to it. This step should only take a few minutes. The snapshot volume, which reflects the original volume at the time of the snapshot, is now available for backing up, while the original volume continues to be available for applications and users.

If required, you can make snapshot volumes for several volumes in a disk group at the same time.

See [“Creating multiple snapshots with the vxassist command”](#) on page 249.

4 Use `fsck` (or some utility appropriate for the application running on the volume) to clean the temporary volume’s contents.

For example, you can use this command with a VxFS file system:

```
# fsck -V vxfs /dev/vx/dsk/diskgroup/snapshot
```

The specified device must have a valid entry in the `/etc/filesystems` file.

5 If you require a backup of the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape, or to some other backup medium.

6 When the backup is complete, you have the following choices for what to do with the snapshot volume:

- Reattach some or all of the plexes of the snapshot volume with the original volume.
See [“Reattaching a snapshot volume”](#) on page 250.
- If FastResync was enabled on the volume before the snapshot was taken, this speeds resynchronization of the snapshot plexes before the backup cycle starts again at step 3.
- Dissociate the snapshot volume entirely from the original volume
See [“Dissociating a snapshot volume”](#) on page 252.
- This may be useful if you want to use the copy for other purposes such as testing or report generation.

- Remove the snapshot volume to save space with this command:

```
# vxedit [-g diskgroup] -rf rm snapshot
```

Dissociating or removing the snapshot volume loses the advantage of fast resynchronization if FastResync was enabled. If there are no further snapshot plexes available, any subsequent snapshots that you take require another complete copy of the original volume to be made.

Converting a plex into a snapshot plex

Note: A plex cannot be converted into a snapshot plex for layered volumes or for any volume that has an associated version 20 DCO volume.

It is recommended that the instant snapshot feature is used in preference to converting a plex into a snapshot plex.

In some circumstances, you may find it more convenient to convert an existing plex in a volume into a snapshot plex rather than running `vxassist snapstart`.

For example, you may want to do this if you are short of disk space for creating the snapshot plex and the volume that you want to snapshot contains more than two plexes.

The procedure can also be used to speed up the creation of a snapshot volume when a mirrored volume is created with more than two plexes and `init=active` is specified.

It is advisable to retain at least two plexes in a volume to maintain data redundancy.

To convert an existing plex into a snapshot plex for a volume on which Persistent FastResync is enabled, use the following command:

```
# vxplex [-g diskgroup] -o dcoplex=dcologplex convert \  
state=SNAPDONE plex
```

`dcologplex` is the name of an existing DCO plex that is to be associated with the new snapshot plex. You can use the `vxprint` command to find out the name of the DCO volume.

See [“Adding a version 0 DCO and DCO volume”](#) on page 253.

For example, to make a snapshot plex from the plex `trivol-03` in the 3-plex volume `trivol`, you would use the following command:

```
# vxplex -o dcoplex=trivol_dco-03 convert state=SNAPDONE \
  trivol-03
```

Here the DCO plex `trivol_dco_03` is specified as the DCO plex for the new snapshot plex.

To convert an existing plex into a snapshot plex in the SNAPDONE state for a volume on which Non-Persistent FastResync is enabled, use the following command:

```
# vxplex [-g diskgroup] convert state=SNAPDONE plex
```

A converted plex is in the SNAPDONE state, and can be used immediately to create a snapshot volume.

Note: The last complete regular plex in a volume, an incomplete regular plex, or a dirty region logging (DRL) log plex cannot be converted into a snapshot plex.

See [“Emulation of third-mirror break-off snapshots”](#) on page 205.

Creating multiple snapshots with the vxassist command

To make it easier to create snapshots of several volumes at the same time, the snapshot option accepts more than one volume name as its argument.

For example:

```
# vxassist [-g diskgroup] snapshot volume1
  volume2 ...
```

By default, the first snapshot volume is named *SNAP-volume*, and each subsequent snapshot is named *SNAPnumber-volume*, where *number* is a unique serial number, and *volume* is the name of the volume for which the snapshot is being taken. This default pattern can be overridden by using the option `-o name=pattern`, as described on the `vxassist(1M)` manual page.

For example, the pattern `SNAP%v-%d` reverses the order of the *number* and *volume* components in the name.

To snapshot all the volumes in a single disk group, specify the option `-o allvols` to `vxassist`:

```
# vxassist -g diskgroup -o allvols snapshot
```

This operation requires that all `vxassist` operations are complete on the volumes. It fails if any of the volumes in the disk group do not have a complete snapshot plex in the SNAPDONE state.

Note: The `vxsnap` command provides similar functionality for creating multiple snapshots.

Reattaching a snapshot volume

The `snapback` operation merges a snapshot copy of a volume with its original. One or more snapshot plexes are detached from the snapshot volume and re-attached to the original volume. The snapshot volume is removed if all its snapshot plexes are snapped back. This task resynchronizes the data in the volume so that the plexes are consistent.

The `snapback` operation cannot be applied to RAID-5 volumes unless they have been converted to a special layered volume layout by the addition of a DCO and DCO volume.

See [“Adding a version 0 DCO and DCO volume”](#) on page 253.

To enhance the efficiency of the `snapback` operation, enable `FastResync` on the volume before taking the snapshot

To merge one snapshot plex with the original volume, use the following command:

```
# vxassist [-g diskgroup] snapback snapshot
```

where *snapshot* is the snapshot copy of the volume.

To merge all snapshot plexes in the snapshot volume with the original volume, use the following command:

```
# vxassist [-g diskgroup] -o allplexes snapback snapshot
```

To merge a specified number of plexes from the snapshot volume with the original volume, use the following command:

```
# vxassist [-g diskgroup] snapback nmirror=number  
snapshot
```

Here the `nmirror` attribute specifies the number of mirrors in the snapshot volume that are to be re-attached.

Once the snapshot plexes have been reattached and their data resynchronized, they are ready to be used in another `snapshot` operation.

By default, the data in the original volume is used to update the snapshot plexes that have been re-attached. To copy the data from the replica volume instead, use the following command:

```
# vxassist [-g diskgroup] -o resyncfromreplica snapback snapshot
```

Warning: Always unmount the snapshot volume (if this is mounted) before performing a snapback. In addition, you must unmount the file system corresponding to the primary volume before using the `resyncfromreplica` option.

Adding plexes to a snapshot volume

If you want to retain the existing plexes in a snapshot volume after a snapback operation, you can create additional snapshot plexes that are to be used for the snapback.

To add plexes to a snapshot volume

- 1 Use the following `vxprint` commands to discover the names of the snapshot volume's data change object (DCO) and DCO volume:

```
# DCONAME=`vxprint [-g diskgroup] -F%dco_name snapshot`  
# DCOVOL=`vxprint [-g diskgroup] -F%log_vol $DCONAME`
```

- 2 Use the `vxassist mirror` command to create mirrors of the existing snapshot volume and its DCO volume:

```
# vxassist -g diskgroup mirror snapshot  
# vxassist -g diskgroup mirror $DCOVOL
```

The new plex in the DCO volume is required for use with the new data plex in the snapshot.

- 3 Use the `vxprint` command to find out the name of the additional snapshot plex:

```
# vxprint -g diskgroup  
    snapshot
```

- 4 Use the `vxprint` command to find out the record ID of the additional DCO plex:

```
# vxprint -g diskgroup -F%rid $DCOVOL
```

- 5 Use the `vxedit` command to set the `dco_plex_rid` field of the new data plex to the name of the new DCO plex:

```
# vxedit -g diskgroup set dco_plex_rid=dco_plex_rid  
    new_plex
```

The new data plex is now ready to be used to perform a snapback operation.

Dissociating a snapshot volume

The link between a snapshot and its original volume can be permanently broken so that the snapshot volume becomes an independent volume. Use the following command to dissociate the snapshot volume, *snapshot*:

```
# vxassist snapclear snapshot
```

Displaying snapshot information

The `vxassist snapprint` command displays the associations between the original volumes and their respective replicas (snapshot copies):

```
# vxassist snapprint [volume]
```

Output from this command is shown in the following examples:

```
# vxassist -g mydg snapprint v1
```

V	NAME	USETYPE	LENGTH	
SS	SNAPOBJ	NAME	LENGTH	%DIRTY
DP	NAME	VOLUME	LENGTH	%DIRTY
v	v1	fsgen	20480	
ss	SNAP-v1_snp	SNAP-v1	20480	4
dp	v1-01	v1	20480	0
dp	v1-02	v1	20480	0
v	SNAP-v1	fsgen	20480	
ss	v1_snp	v1	20480	0

```
# vxassist -g mydg snapprint v2
```

V	NAME	USETYPE	LENGTH	
SS	SNAPOBJ	NAME	LENGTH	%DIRTY
DP	NAME	VOLUME	LENGTH	%DIRTY
v	v2	fsgen	20480	
ss	--	SNAP-v2	20480	0
dp	v2-01	v2	20480	0
v	SNAP-v2	fsgen	20480	
ss	--	v2	20480	0

In this example, Persistent FastResync is enabled on volume `v1`, and Non-Persistent FastResync on volume `v2`. Lines beginning with `v`, `dp` and `ss` indicate a volume, detached plex and snapshot plex respectively. The `%DIRTY` field indicates the percentage of a snapshot plex or detached plex that is dirty with respect to the original volume. Notice that no snap objects are associated with volume `v2` or with its snapshot volume `SNAP-v2`.

If a volume is specified, the `snapprint` command displays an error message if no FastResync maps are enabled for that volume.

Adding a version 0 DCO and DCO volume

The version 0 DCO log volume was introduced in VxVM 3.2. The version 0 layout supports traditional (third-mirror break-off) snapshots, but not full-sized or space-optimized instant snapshots.

To put Persistent FastResync into effect for a volume, a Data Change Object (DCO) and DCO volume must first be associated with that volume. When you have added a DCO object and DCO volume to a volume, you can then enable Persistent FastResync on the volume

Note: You need a FastResync license key to use the FastResync feature. Even if you do not have a license, you can configure a DCO object and DCO volume so that snap objects are associated with the original and snapshot volumes.

To add a DCO object and DCO volume to an existing volume

- 1 Ensure that the disk group containing the existing volume has been upgraded to at least version 90. Use the following command to check the version of a disk group:

```
# vxdg list diskgroup
```

To upgrade a disk group to the latest version, use the following command:

```
# vxdg upgrade diskgroup
```

For more information about disk group versions, see the *Veritas Volume Manager Administrator's Guide*.

- 2 Use the following command to turn off Non-Persistent FastResync on the original volume if it is currently enabled:

```
# vxvol [-g diskgroup] set fastresync=off volume
```

If you are uncertain about which volumes have Non-Persistent FastResync enabled, use the following command to obtain a listing of such volumes.

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

```
# vxprint [-g diskgroup] -F "%name" \  
-e "v_fastresync=on && \!v_hasdcolog"
```

Use the following command to add a DCO and DCO volume to the existing volume (which may already have dirty region logging (DRL) enabled):

```
# vxassist [-g diskgroup] addlog volume logtype=dc0 \  
[ndcomirror=number] [dcolen=size] [storage_attributes]
```

For non-layered volumes, the default number of plexes in the mirrored DCO volume is equal to the lesser of the number of plexes in the data volume or 2. For layered volumes, the default number of DCO plexes is always 2. If required, use the `ndcomirror` attribute to specify a different number. It is recommended that you configure as many DCO plexes as there are existing data and snapshot plexes in the volume.

For example, specify `ndcomirror=3` when adding a DCO to a 3-way mirrored volume.

The default size of each plex is 132 blocks. You can use the `dcolen` attribute to specify a different *size*. If specified, the size of the plex must be an integer multiple of 33 blocks from 33 up to a maximum of 2112 blocks.

You can specify `vxassist`-style storage attributes to define the disks that can and/or cannot be used for the plexes of the DCO volume.

See [“Specifying storage for version 0 DCO plexes”](#) on page 255.

Specifying storage for version 0 DCO plexes

If the disks that contain volumes and their snapshots are to be moved or split into different disk groups, the disks that contain their respective DCO plexes must be able to accompany them. By default, VxVM attempts to place version 0 DCO plexes on the same disks as the data plexes of the parent volume. However, this may be impossible if there is insufficient space available on those disks. In this case, VxVM uses any available space on other disks in the disk group. If the DCO plexes are

placed on disks which are used to hold the plexes of other volumes, this may cause problems when you subsequently attempt to move volumes into other disk groups.

You can use storage attributes to specify explicitly which disks to use for the DCO plexes. If possible, specify the same disks as those on which the volume is configured.

For example, to add a DCO object and DCO volume with plexes on `mydg05` and `mydg06`, and a plex size of 264 blocks to the volume, `myvol`, in the disk group, `mydg`, use the following command:

```
# vxassist -g mydg addlog myvol logtype=dco dcolen=264 mydg05 mydg06
```

To view the details of the DCO object and DCO volume that are associated with a volume, use the `vxprint` command. The following is partial `vxprint` output for the volume named `vol1` (the `TUTIL0` and `PUTILO` columns are omitted for clarity):

TY	NAME	ASSOC	KSTATE	LENGTH	PLOFFS	STATE	...
v	vol1	fsgen	ENABLED	1024	-	ACTIVE	
pl	vol1-01	vol1	ENABLED	1024	-	ACTIVE	
sd	disk01-01	vol1-01	ENABLED	1024	0	-	
pl	vol1-02	vol1	ENABLED	1024	-	ACTIVE	
sd	disk02-01	vol1-02	ENABLED	1024	0	-	
dc	vol1_dco	vol1	-	-	-	-	
v	vol1_dcl	gen	ENABLED	132	-	ACTIVE	
pl	vol1_dcl-01	vol1_dcl	ENABLED	132	-	ACTIVE	
sd	disk03-01	vol1_dcl-01	ENABLED	132	0	-	
pl	vol1_dcl-02	vol1_dcl	ENABLED	132	-	ACTIVE	
sd	disk04-01	vol1_dcl-02	ENABLED	132	0	-	

In this output, the DCO object is shown as `vol1_dco`, and the DCO volume as `vol1_dcl` with 2 plexes, `vol1_dcl-01` and `vol1_dcl-02`.

If required, you can use the `vxassist move` command to relocate DCO plexes to different disks.

For example, the following command moves the plexes of the DCO volume, `vol1_dcl`, for volume `vol1` from `disk03` and `disk04` to `disk07` and `disk08`.

The `!` character is a special character in some shells. The following example shows how to escape it in a bash shell.

```
# vxassist -g mydg move vol1_dcl \!disk03 \!disk04 disk07 disk08
```

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

Removing a version 0 DCO and DCO volume

To dissociate a version 0 DCO object, DCO volume and any snap objects from a volume, use the following command:

```
# vxassist [-g diskgroup] remove log volume logtype=dcv
```

This completely removes the DCO object, DCO volume and any snap objects. It also has the effect of disabling FastResync for the volume.

Alternatively, you can use the `vxdco` command to the same effect:

```
# vxdco [-g diskgroup] [-o rm] dis dco_obj
```

The default name of the DCO object, `dco_obj`, for a volume is usually formed by appending the string `_dco` to the name of the parent volume. To find out the name of the associated DCO object, use the `vxprint` command on the volume.

To dissociate, but not remove, the DCO object, DCO volume and any snap objects from the volume, `myvol`, in the disk group, `mydg`, use the following command:

```
# vxdco -g mydg dis myvol_dco
```

This form of the command dissociates the DCO object from the volume but does not destroy it or the DCO volume. If the `-o rm` option is specified, the DCO object, DCO volume and its plexes, and any snap objects are also removed.

Warning: Dissociating a DCO and DCO volume disables Persistent FastResync on the volume. A full resynchronization of any remaining snapshots is required when they are snapped back.

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

See the `vxdco(1M)` manual pages.

Reattaching a version 0 DCO and DCO volume

If a version 0 DCO object and DCO volume are not removed by specifying the `-o rm` option to `vxdco`, they can be reattached to the parent volume using the following command:

```
# vxdco [-g diskgroup] att volume
    dco_obj
```

For example, to reattach the DCO object, `myvol_dco`, to the volume, `myvol`, use the following command:

```
# vxdco -g mydg att myvol myvol_dco
```

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

Administering snapshot file systems

This chapter includes the following topics:

- [About snapshot file systems](#)
- [How a snapshot file system works](#)
- [Snapshot file system backups](#)
- [Snapshot file system performance](#)
- [About snapshot file system disk structure](#)
- [Differences between snapshots and Storage Checkpoints](#)
- [Creating a snapshot file system](#)
- [Backup examples](#)

About snapshot file systems

A snapshot file system is an exact image of a VxFS file system, referred to as the snapped file system, that provides a mechanism for making backups. The snapshot is a consistent view of the file system “snapped” at the point in time the snapshot is made. You can select files to back up from the snapshot using a standard utility such as `cpio` or `cp`, or back up the entire file system image using the `vxdump` or `fscat` utilities.

You use the `mount` command to create a snapshot file system; the `mkfs` command is not required. A snapshot file system is always read-only. A snapshot file system exists only as long as the snapped file system is mounted, and the snapshot file system ceases to exist when unmounted. A snapped file system cannot be

unmounted until all of its snapshots are unmounted. Although it is possible to have multiple snapshots of a file system made at different times, it is not possible to make a snapshot of a snapshot.

Note: A snapshot file system ceases to exist when unmounted. If mounted again, it is actually a fresh snapshot of the snapped file system. A snapshot file system must be unmounted before its dependent snapped file system can be unmounted. Neither the `fuser` command nor the `mount` command will indicate that a snapped file system cannot be unmounted because a snapshot of it exists.

On cluster file systems, snapshots can be created on any node in the cluster, and backup operations can be performed from that node. The snapshot of a cluster file system is accessible only on the node where it is created, that is, the snapshot file system itself cannot be cluster mounted.

See the *Veritas Storage Foundation Cluster File System Administrator's Guide*.

How a snapshot file system works

A snapshot file system is created by mounting an empty disk slice as a snapshot of a currently mounted file system. The bitmap, blockmap and super-block are initialized and then the currently mounted file system is frozen. After the file system to be snapped is frozen, the snapshot is enabled and mounted and the snapped file system is thawed. The snapshot appears as an exact image of the snapped file system at the time the snapshot was made.

Initially, the snapshot file system satisfies read requests by finding the data on the snapped file system and returning it to the requesting process. When an inode update or a write changes the data in block *n* of the snapped file system, the old data is first read and copied to the snapshot before the snapped file system is updated. The bitmap entry for block *n* is changed from 0 to 1, indicating that the data for block *n* can be found on the snapshot file system. The blockmap entry for block *n* is changed from 0 to the block number on the snapshot file system containing the old data.

A subsequent read request for block *n* on the snapshot file system will be satisfied by checking the bitmap entry for block *n* and reading the data from the indicated block on the snapshot file system, instead of from block *n* on the snapped file system. This technique is called copy-on-write. Subsequent writes to block *n* on the snapped file system do not result in additional copies to the snapshot file system, since the old data only needs to be saved once.

All updates to the snapped file system for inodes, directories, data in files, extent maps, and so forth, are handled in this fashion so that the snapshot can present

a consistent view of all file system structures on the snapped file system for the time when the snapshot was created. As data blocks are changed on the snapped file system, the snapshot gradually fills with data copied from the snapped file system.

The amount of disk space required for the snapshot depends on the rate of change of the snapped file system and the amount of time the snapshot is maintained. In the worst case, the snapped file system is completely full and every file is removed and rewritten. The snapshot file system would need enough blocks to hold a copy of every block on the snapped file system, plus additional blocks for the data structures that make up the snapshot file system. This is approximately 101 percent of the size of the snapped file system. Normally, most file systems do not undergo changes at this extreme rate. During periods of low activity, the snapshot should only require two to six percent of the blocks of the snapped file system. During periods of high activity, the snapshot might require 15 percent of the blocks of the snapped file system. These percentages tend to be lower for larger file systems and higher for smaller ones.

Warning: If a snapshot file system runs out of space for changed data blocks, it is disabled and all further attempts to access it fails. This does not affect the snapped file system.

Snapshot file system backups

After a snapshot file system is created, the snapshot maintains a consistent backup of data in the snapped file system.

Backup programs, such as `cpio`, that back up a standard file system tree can be used without modification on a snapshot file system because the snapshot presents the same data as the snapped file system. Backup programs, such as `vxdump`, that access the disk structures of a file system require some modifications to handle a snapshot file system.

VxFS utilities recognize snapshot file systems and modify their behavior so that they operate the same way on snapshots as they do on standard file systems. Other backup programs that typically read the raw disk image cannot work on snapshots without altering the backup procedure.

These other backup programs can use the `fscat` command to obtain a raw image of the entire file system that is identical to an image obtainable by running a `dd` command on the disk device containing the snapped file system at the exact moment the snapshot was created. The `snapread ioctl` takes arguments similar to those of the `read` system call and returns the same results that are obtainable by performing a read on the disk device containing the snapped file system at the

exact time the snapshot was created. In both cases, however, the snapshot file system provides a consistent image of the snapped file system with all activity complete—it is an instantaneous read of the entire file system. This is much different than the results that would be obtained by a `dd` or `read` command on the disk device of an active file system.

Snapshot file system performance

Snapshot file systems maximize the performance of the snapshot at the expense of writes to the snapped file system. Reads from a snapshot file system typically perform at nearly the throughput rates of reads from a standard VxFS file system.

The performance of reads from the snapped file system are generally not affected. However, writes to the snapped file system, typically average two to three times as long as without a snapshot. This is because the initial write to a data block requires reading the old data, writing the data to the snapshot, and then writing the new data to the snapped file system. If there are multiple snapshots of the same snapped file system, writes are even slower. Only the initial write to a block experiences this delay, so operations such as writes to the intent log or inode updates proceed at normal speed after the initial write.

Reads from the snapshot file system are impacted if the snapped file system is busy because the snapshot reads are slowed by the disk I/O associated with the snapped file system.

The overall impact of the snapshot is dependent on the read to write ratio of an application and the mixing of the I/O operations. For example, a database application running an online transaction processing (OLTP) workload on a snapped file system was measured at about 15 to 20 percent slower than a file system that was not snapped.

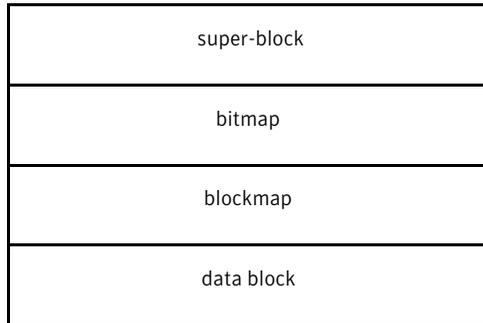
About snapshot file system disk structure

A snapshot file system consists of:

- A super-block
- A bitmap
- A blockmap
- Data blocks copied from the snapped file system

The following figure shows the disk structure of a snapshot file system.

Figure 18-1 The Snapshot Disk Structure



The super-block is similar to the super-block of a standard VxFS file system, but the magic number is different and many of the fields are not applicable.

The bitmap contains one bit for every block on the snapped file system. Initially, all bitmap entries are zero. A set bit indicates that the appropriate block was copied from the snapped file system to the snapshot. In this case, the appropriate position in the blockmap references the copied block.

The blockmap contains one entry for each block on the snapped file system. Initially, all entries are zero. When a block is copied from the snapped file system to the snapshot, the appropriate entry in the blockmap is changed to contain the block number on the snapshot file system that holds the data from the snapped file system.

The data blocks are filled by data copied from the snapped file system, starting from the beginning of the data block area.

Differences between snapshots and Storage Checkpoints

While snapshots and Storage Checkpoints both create a point-in-time image of a file system and only the changed data blocks are updated, there are significant differences between the two technologies:

Table 18-1 Differences between snapshots and Storage Checkpoints

Snapshots	Storage Checkpoints
Require a separate device for storage	Reside on the same device as the original file system
Are read-only	Can be read-only or read-write

Table 18-1 Differences between snapshots and Storage Checkpoints (*continued*)

Snapshots	Storage Checkpoints
Are transient	Are persistent
Cease to exist after being unmounted	Can exist and be mounted on their own
Track changed blocks on the file system level	Track changed blocks on each file in the file system

Storage Checkpoints also serve as the enabling technology for two other Veritas features: Block-Level Incremental Backups and Storage Rollback, which are used extensively for backing up databases.

See [“About Storage Checkpoints”](#) on page 267.

Creating a snapshot file system

You create a snapshot file system by using the `-o snapof=` option of the `mount` command. The `-o snapsize=` option may also be required if the device you are mounting does not identify the device size in its disk label, or if you want a size smaller than the entire device.

You must make the snapshot file system large enough to hold any blocks on the snapped file system that may be written to while the snapshot file system exists. If a snapshot runs out of blocks to hold copied data, the snapshot is disabled and further attempts to access the snapshot file system fail.

During periods of low activity (such as nights and weekends), a snapshot typically requires about two to six percent of the blocks of the snapped file system. During a period of high activity, the snapshot of a typical file system may require 15 percent of the blocks of the snapped file system. Most file systems do not turn over 15 percent of data in a single day. These approximate percentages tend to be lower for larger file systems and higher for smaller file systems. You can allocate blocks to a snapshot based on characteristics such as file system usage and duration of backups.

Warning: Any existing data on the device used for the snapshot is overwritten.

To create a snapshot file system

- ◆ Mount the file system with the `-o snapof=` option:

```
# mount -V vxfs -o snapof=special,snapsize=snapshot_size \  
snapshot_special snapshot_mount_point
```

Backup examples

The `vxdump` utility ascertains whether `/dev/rdisk/fsvol/vol1` is a snapshot mounted as `/backup/home` and does the appropriate work to get the snapshot data through the mount point.

The following are typical examples of making a backup of a 300,000 block file system named `/home` using a snapshot file system on a Volume Manager volume with a snapshot mount point of `/backup/home`.

To create a backup using a snapshot file system

- 1 To back up files changed within the last week using `cpio`:

```
# mount -V vxfs -o snapof=/home,snapsize=100000 \  
/dev/vx/dsk/fsvol/vol1 /backup/home  
# cd /backup  
# find home -ctime -7 -depth -print | cpio -oc > /dev/rfd0  
# umount /backup/home
```

- 2 To do a level 3 backup of `/dev/vx/rdsk/fsvol/vol1` and collect those files that have changed in the current directory:

```
# vxdump 3f - /dev/vx/rdsk/fsvol/vol1 | vxrestore -xf -
```

- 3 To do a full backup of `/home`, which exists on disk `/dev/vx/rdsk/fsvol/vol1`, and use `dd` to control blocking of output onto tape device using `vxdump`:

```
# mount -V vxfs -o snapof=/home,snapsize=100000 \  
/dev/vx/dsk/fsvol/vol1 /backup/home  
# vxdump f - /dev/vx/rdsk/fsvol/vol1 | dd bs=128k > \  
/dev/rfd0
```


Administering Storage Checkpoints

This chapter includes the following topics:

- [About Storage Checkpoints](#)
- [Operation of a Storage Checkpoint](#)
- [Types of Storage Checkpoints](#)
- [Storage Checkpoint administration](#)
- [Storage Checkpoint space management considerations](#)
- [Restoring from a Storage Checkpoint](#)
- [Storage Checkpoint quotas](#)

About Storage Checkpoints

Veritas File System (VxFS) provides a Storage Checkpoint feature that quickly creates a persistent image of a file system at an exact point in time. Storage Checkpoints significantly reduce I/O overhead by identifying and maintaining only the file system blocks that have changed since the last Storage Checkpoint or backup via a copy-on-write technique.

See “[Copy-on-write](#)” on page 271.

Storage Checkpoints provide:

- Persistence through reboots and crashes.
- The ability for data to be immediately writeable by preserving the file system metadata, the directory hierarchy, and user data.

Storage Checkpoints are actually data objects that are managed and controlled by the file system. You can create, remove, and rename Storage Checkpoints because they are data objects with associated names.

See [“Operation of a Storage Checkpoint”](#) on page 269.

Unlike a disk-based mirroring technology that requires a separate storage space, Storage Checkpoints minimize the use of disk space by using a Storage Checkpoint within the same free space available to the file system.

After you create a Storage Checkpoint of a mounted file system, you can also continue to create, remove, and update files on the file system without affecting the logical image of the Storage Checkpoint. A Storage Checkpoint preserves not only the name space (directory hierarchy) of the file system, but also the user data as it existed at the moment the file system image was captured.

You can use a Storage checkpoint in many ways. For example, you can use them to:

- Create a stable image of the file system that can be backed up to tape.
- Provide a mounted, on-disk backup of the file system so that end users can restore their own files in the event of accidental deletion. This is especially useful in a home directory, engineering, or email environment.
- Create a copy of an application's binaries before installing a patch to allow for rollback in case of problems.
- Create an on-disk backup of the file system in that can be used in addition to a traditional tape-based backup to provide faster backup and restore capabilities.
- Test new software on a point-in-time image of the primary fileset without jeopardizing the live data in the current primary fileset by mounting the Storage Checkpoints as writable.

Distinguishing between Storage Checkpoints and snapshots

Storage Checkpoints differ from Veritas File System snapshots in the following ways because they:

- Allow write operations to the Storage Checkpoint itself.
- Persist after a system reboot or failure.
- Share the same pool of free space as the file system.
- Maintain a relationship with other Storage Checkpoints by identifying changed file blocks since the last Storage Checkpoint.

- Can have multiple, read-only Storage Checkpoints that reduce I/O operations and required storage space because the most recent Storage Checkpoint is the only one that accumulates updates from the primary file system.
- Can restore the file system to its state at the time that the Storage Checkpoint was taken.

Various backup and replication solutions can take advantage of Storage Checkpoints. The ability of Storage Checkpoints to track the file system blocks that have changed since the last Storage Checkpoint facilitates backup and replication applications that only need to retrieve the changed data. Storage Checkpoints significantly minimize data movement and may promote higher availability and data integrity by increasing the frequency of backup and replication solutions.

Storage Checkpoints can be taken in environments with a large number of files, such as file servers with millions of files, with little adverse impact on performance. Because the file system does not remain frozen during Storage Checkpoint creation, applications can access the file system even while the Storage Checkpoint is taken. However, Storage Checkpoint creation may take several minutes to complete depending on the number of files in the file system.

Operation of a Storage Checkpoint

The Storage Checkpoint facility freezes the mounted file system (known as the primary fileset), initializes the Storage Checkpoint, and thaws the file system. Specifically, the file system is first brought to a stable state where all of its data is written to disk, and the freezing process momentarily blocks all I/O operations to the file system. A Storage Checkpoint is then created without any actual data; the Storage Checkpoint instead points to the block map of the primary fileset. The thawing process that follows restarts I/O operations to the file system.

You can create a Storage Checkpoint on a single file system or a list of file systems. A Storage Checkpoint of multiple file systems simultaneously freezes the file systems, creates a Storage Checkpoint on all of the file systems, and thaws the file systems. As a result, the Storage Checkpoints for multiple file systems have the same creation timestamp. The Storage Checkpoint facility guarantees that multiple file system Storage Checkpoints are created on all or none of the specified file systems, unless there is a system crash while the operation is in progress.

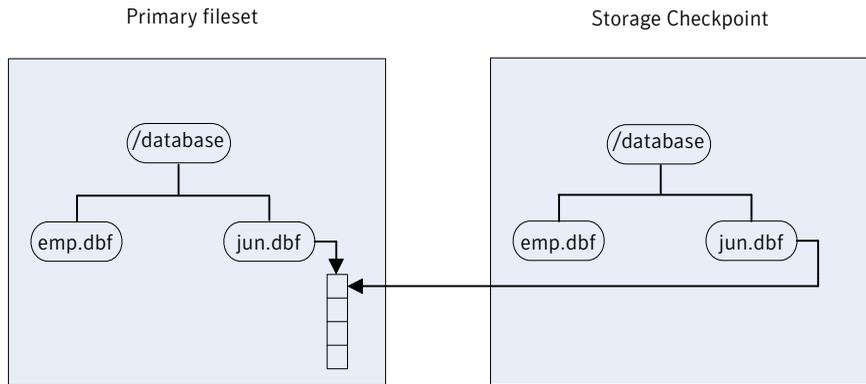
Note: The calling application is responsible for cleaning up Storage Checkpoints after a system crash.

A Storage Checkpoint of the primary fileset initially contains only pointers to the existing data blocks in the primary fileset, and does not contain any allocated data blocks of its own.

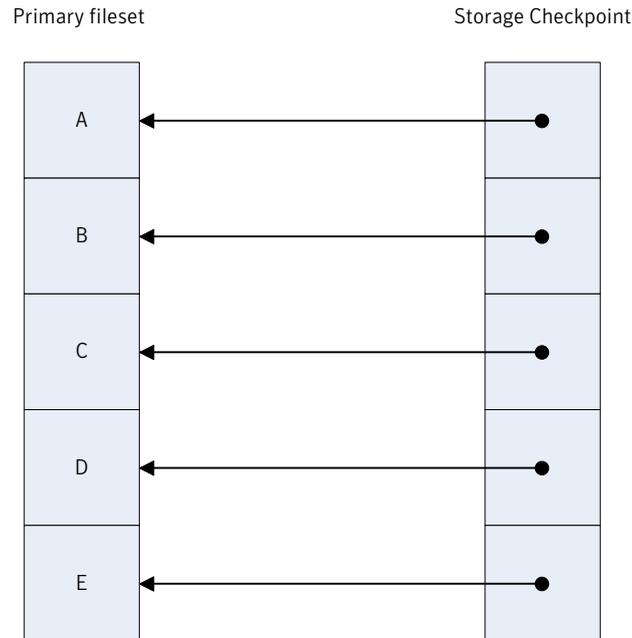
Figure 19-1 shows the file system `/database` and its Storage Checkpoint.

The Storage Checkpoint is logically identical to the primary fileset when the Storage Checkpoint is created, but it does not contain any actual data blocks.

Figure 19-1 Primary fileset and its Storage Checkpoint



In Figure 19-2, a square represents each block of the file system. This figure shows a Storage Checkpoint containing pointers to the primary fileset at the time the Storage Checkpoint is taken, as in Figure 19-1.

Figure 19-2 Initializing a Storage Checkpoint

The Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. VxFS updates a Storage Checkpoint by using the copy-on-write technique.

See [“Copy-on-write”](#) on page 271.

Copy-on-write

In [Figure 19-3](#), the third data block in the primary fileset originally containing C is updated.

Before the data block is updated with new data, the original data is copied to the Storage Checkpoint. This is called the copy-on-write technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken.

Every update or write operation does not necessarily result in the process of copying data to the Storage Checkpoint because the old data needs to be saved only once. As blocks in the primary fileset continue to change, the Storage Checkpoint accumulates the original data blocks. In this example, subsequent updates to the third data block, now containing C', are not copied to the Storage Checkpoint because the original image of the block containing C is already saved.

limit the life of data Storage Checkpoints to minimize the impact on system resources.

See [“Showing the difference between a data and a nodata Storage Checkpoint”](#) on page 279.

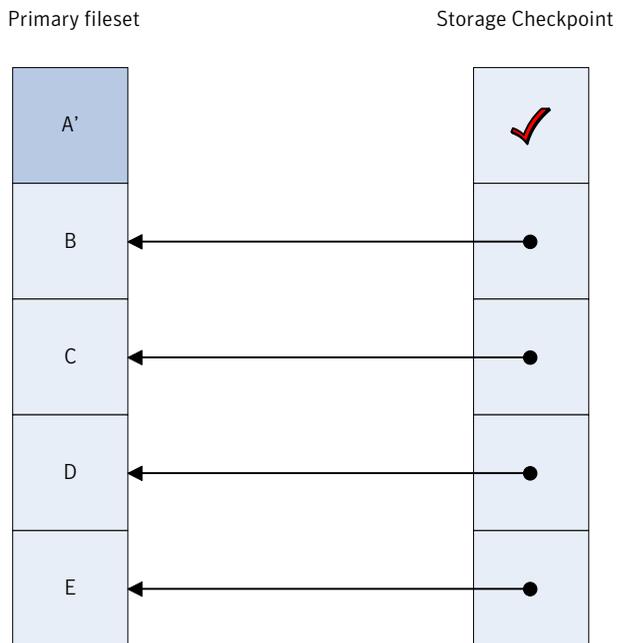
Nodata Storage Checkpoints

A nodata Storage Checkpoint only contains file system metadata—no file data blocks. As the original file system changes, the nodata Storage Checkpoint records the location of every changed block. Nodata Storage Checkpoints use minimal system resources and have little impact on the performance of the file system because the data itself does not have to be copied.

In [Figure 19-4](#), the first block originally containing A is updated.

The original data is not copied to the storage checkpoint, but the changed block is marked in the Storage Checkpoint. The marker indicates which data has changed.

Figure 19-4 Updates to a nodata clone



See [“Showing the difference between a data and a nodata Storage Checkpoint”](#) on page 279.

Removable Storage Checkpoints

A removable Storage Checkpoint can “self-destruct” under certain conditions when the file system runs out of space.

See “[Storage Checkpoint space management considerations](#)” on page 286.

After encountering certain out-of-space (`ENOSPC`) conditions, the kernel removes Storage Checkpoints to free up space for the application to continue running on the file system. In almost all situations, you should create Storage Checkpoints with the removable attribute.

Non-mountable Storage Checkpoints

You can create Storage Checkpoints that cannot be mounted by using the `fsckptadm set nomount` command.

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

Use this type of Storage Checkpoint as a security feature which prevents other applications from accessing the Storage Checkpoint and modifying it.

Storage Checkpoint administration

Storage Checkpoint administrative operations require the `fsckptadm` utility.

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

You can use the `fsckptadm` utility to create and remove Storage Checkpoints, change attributes, and ascertain statistical data. Every Storage Checkpoint has an associated name, which allows you to manage Storage Checkpoints; this name is limited to 127 characters and cannot contain a colon (:).

Storage Checkpoints require some space for metadata on the volume or set of volumes specified by the file system allocation policy or Storage Checkpoint allocation policy. The `fsckptadm` utility displays an error if the volume or set of volumes does not have enough free space to contain the metadata. You can roughly approximate the amount of space required by the metadata using a method that depends on the disk layout version of the file system.

For disk layout Version 6 or 7, multiply the number of inodes by 1 byte, and add 1 or 2 megabytes to get the approximate amount of space required. You can determine the number of inodes with the `fsckptadm` utility.

Use the `fsvoladm` command to determine if the volume set has enough free space.

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

```
# fsvoladm list /mnt0
dev    size      used      avail      name
0      20971520    8497658   12473862   mnt1
1      20971520    6328993   14642527   mnt2
2      20971520    4458462   16513058   mnt3
```

Creating a Storage Checkpoint

The following example shows the creation of a nodata Storage Checkpoint named `thu_7pm` on `/mnt0` and lists all Storage Checkpoints of the `/mnt0` file system:

```
# fsckptadm -n create thu_7pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_7pm:
  ctime   = Thu 3 Mar 2005 7:00:17 PM PST
  mtime   = Thu 3 Mar 2005 7:00:17 PM PST
  flags   = nodata, largefiles
```

The following example shows the creation of a removable Storage Checkpoint named `thu_8pm` on `/mnt0` and lists all Storage Checkpoints of the `/mnt0` file system:

```
# fsckptadm -r create thu_8pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_8pm:
  ctime   = Thu 3 Mar 2005 8:00:19 PM PST
  mtime   = Thu 3 Mar 2005 8:00:19 PM PST
  flags   = largefiles, removable
thu_7pm:
  ctime   = Thu 3 Mar 2005 7:00:17 PM PST
  mtime   = Thu 3 Mar 2005 7:00:17 PM PST
  flags   = nodata, largefiles
```

Removing a Storage Checkpoint

You can delete a Storage Checkpoint by specifying the remove keyword of the `fsckptadm` command. Specifically, you can use either the synchronous or asynchronous method of removing a Storage Checkpoint; the asynchronous method is the default method. The synchronous method entirely removes the Storage Checkpoint and returns all of the blocks to the file system before completing the `fsckptadm` operation. The asynchronous method simply marks the Storage Checkpoint for removal and causes `fsckptadm` to return immediately.

At a later time, an independent kernel thread completes the removal operation and releases the space used by the Storage Checkpoint.

In this example, `/mnt0` is a mounted VxFS file system with a Version 6 disk layout. This example shows the asynchronous removal of the Storage Checkpoint named `thu_8pm` and synchronous removal of the Storage Checkpoint named `thu_7pm`. This example also lists all the Storage Checkpoints remaining on the `/mnt0` file system after the specified Storage Checkpoint is removed:

```
# fsckptadm remove thu_8pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_7pm:
  ctime   = Thu 3 Mar 2005 7:00:17 PM PST
  mtime   = Thu 3 Mar 2005 7:00:17 PM PST
  flags   = nodata, largefiles
# fsckptadm -s remove thu_7pm /mnt0
# fsckptadm list /mnt0
/mnt0
```

Accessing a Storage Checkpoint

You can mount Storage Checkpoints using the `mount` command with the `mount` option `-o ckpt=ckpt_name`.

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

Observe the following rules when mounting Storage Checkpoints:

- Storage Checkpoints are mounted as read-only Storage Checkpoints by default. If you must write to a Storage Checkpoint, mount it using the `-o rw` option.
- If a Storage Checkpoint is currently mounted as a read-only Storage Checkpoint, you can remount it as a writable Storage Checkpoint using the `-o remount` option.
- To mount a Storage Checkpoint of a file system, first mount the file system itself.
- To unmount a file system, first unmount all of its Storage Checkpoints.

Warning: If you create a Storage Checkpoint for backup purposes, do not mount it as a writable Storage Checkpoint. You will lose the point-in-time image if you accidentally write to the Storage Checkpoint.

If older Storage Checkpoints already exist, write activity to a writable Storage Checkpoint can generate copy operations and increased space usage in the older Storage Checkpoints.

A Storage Checkpoint is mounted on a special pseudo device. This pseudo device does not exist in the system name space; the device is internally created by the system and used while the Storage Checkpoint is mounted. The pseudo device is removed after you unmount the Storage Checkpoint. A pseudo device name is formed by appending the Storage Checkpoint name to the file system device name using the colon character (:) as the separator.

For example, if a Storage Checkpoint named `may_23` belongs to the file system residing on the special device `/dev/vx/dsk/fsvol/vol1`, the Storage Checkpoint pseudo device name is:

```
/dev/vx/dsk/fsvol/vol1:may_23
```

- To mount the Storage Checkpoint named `may_23` as a read-only Storage Checkpoint on directory `/fsvol_may_23`, type:

```
# mount -V vxfs -o ckpt=may_23 /dev/vx/dsk/fsvol/vol1:may_23 \  
/fsvol_may_23
```

Note: The `vol1` file system must already be mounted before the Storage Checkpoint can be mounted.

- To remount the Storage Checkpoint named `may_23` as a writable Storage Checkpoint, type:

```
# mount -V vxfs -o ckpt=may_23,remount,rw \  
/dev/vx/dsk/fsvol/vol1:may_23 /fsvol_may_23
```

- To mount this Storage Checkpoint automatically when the system starts up, put the following entries in the `/etc/filesystems` file:

```
/fsvol
```

```
dev          = /dev/vx/dsk/fsvol/vol1
```

```
vfs          = vxfs
```

```

                                mount      = automatic
                                check      = 1 (or true)

/fsvol_may_23

                                dev        = /dev/vx/dsk/fsvol/vol1:may_23
                                vfs        = vxfs
                                mount      = automatic
                                options    = ckpt=may_23

```

- To mount a Storage Checkpoint of a cluster file system, you must also use the `-o cluster` option:

```
# mount -V vxfs -o cluster,ckpt=may_23 \
/dev/vx/dsk/fsvol/vol1:may_23 /fsvol_may_23
```

You can only mount a Storage Checkpoint cluster-wide if the file system that the Storage Checkpoint belongs to is also mounted cluster-wide. Similarly, you can only mount a Storage Checkpoint locally if the file system that the Storage Checkpoint belongs to is mounted locally.

You can unmount Storage Checkpoints using the `umount` command.

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

Storage Checkpoints can be unmounted by the mount point or pseudo device name:

```
# umount /fsvol_may_23
# umount /dev/vx/dsk/fsvol/vol1:may_23
```

Note: You do not need to run the `fsck` utility on Storage Checkpoint pseudo devices because pseudo devices are part of the actual file system.

Converting a data Storage Checkpoint to a nodata Storage Checkpoint

A nodata Storage Checkpoint does not contain actual file data. Instead, this type of Storage Checkpoint contains a collection of markers indicating the location of all the changed blocks since the Storage Checkpoint was created.

See “[Types of Storage Checkpoints](#)” on page 272.

You can use either the synchronous or asynchronous method to convert a data Storage Checkpoint to a nodata Storage Checkpoint; the asynchronous method

is the default method. In a synchronous conversion, `fsckptadm` waits for all files to undergo the conversion process to “nodata” status before completing the operation. In an asynchronous conversion, `fsckptadm` returns immediately and marks the Storage Checkpoint as a nodata Storage Checkpoint even though the Storage Checkpoint's data blocks are not immediately returned to the pool of free blocks in the file system. The Storage Checkpoint deallocates all of its file data blocks in the background and eventually returns them to the pool of free blocks in the file system.

If all of the older Storage Checkpoints in a file system are nodata Storage Checkpoints, use the synchronous method to convert a data Storage Checkpoint to a nodata Storage Checkpoint. If an older data Storage Checkpoint exists in the file system, use the asynchronous method to mark the Storage Checkpoint you want to convert for a delayed conversion. In this case, the actual conversion will continue to be delayed until the Storage Checkpoint becomes the oldest Storage Checkpoint in the file system, or all of the older Storage Checkpoints have been converted to nodata Storage Checkpoints.

Note: You cannot convert a nodata Storage Checkpoint to a data Storage Checkpoint because a nodata Storage Checkpoint only keeps track of the location of block changes and does not save the content of file data blocks.

Showing the difference between a data and a nodata Storage Checkpoint

The following example shows the difference between data Storage Checkpoints and nodata Storage Checkpoints.

Note: A nodata Storage Checkpoint does not contain actual file data.

To show the difference between Storage Checkpoints

- 1 Create a file system and mount it on `/mnt0`, as in the following example:

```
# mkfs -V vxfs /dev/vx/rdisk/dg1/test0

version 7 layout
134217728 sectors, 67108864 blocks of size 1024, log \

size 65536 blocks, largefiles supported
# mount -V /dev/vx/rdisk/dg1/test0 /mnt0
```

- 2 Create a small file with a known content, as in the following example:

```
# echo "hello, world" > /mnt0/file
```

- 3 Create a Storage Checkpoint and mount it on `/mnt0@5_30pm`, as in the following example:

```
# fsckptadm create ckpt@5_30pm /mnt0
# mkdir /mnt0@5_30pm
# mount -V vxfs -o ckpt=ckpt@5_30pm \
/dev/vx/dsk/dg1/test0:ckpt@5_30pm /mnt0@5_30pm
```

- 4 Examine the content of the original file and the Storage Checkpoint file:

```
# cat /mnt0/file
hello, world
# cat /mnt0@5_30pm/file
hello, world
```

- 5 Change the content of the original file:

```
# echo "goodbye" > /mnt0/file
```

- 6 Examine the content of the original file and the Storage Checkpoint file. The original file contains the latest data while the Storage Checkpoint file still contains the data at the time of the Storage Checkpoint creation:

```
# cat /mnt0/file
goodbye
# cat /mnt0@5_30pm/file
hello, world
```

- 7 Unmount the Storage Checkpoint, convert the Storage Checkpoint to a nodata Storage Checkpoint, and mount the Storage Checkpoint again:

```
# umount /mnt0@5_30pm
# fsckptadm -s set nodata ckpt@5_30pm /mnt0
# mount -V vxfs -o ckpt=ckpt@5_30pm \
/dev/vx/dsk/dg1/test0:ckpt@5_30pm /mnt0@5_30pm
```

- 8 Examine the content of both files. The original file must contain the latest data:

```
# cat /mnt0/file
goodbye
```

You can traverse and read the directories of the nodata Storage Checkpoint; however, the files contain no data, only markers to indicate which block of the file has been changed since the Storage Checkpoint was created:

```
# ls -l /mnt0@5_30pm/file
-rw-r--r-- 1 root other 13 Jul 13 17:13 \ mnt0@5_30pm/file
# cat /mnt0@5_30pm/file
cat: /mnt0@5_30pm/file: I/O error
```

Converting multiple Storage Checkpoints

You can convert Storage Checkpoints to nodata Storage Checkpoints, when dealing with older Storage Checkpoints on the same file system.

To convert multiple Storage Checkpoints

- 1 Create a file system and mount it on /mnt0:

```
# mkfs -V vxfs /dev/vx/rdisk/dg1/test0
version 7 layout
13417728 sectors, 67108864 blocks of size 1024, log \
size 65536 blocks largefiles supported
# mount -V vxfs /dev/vx/dsk/dg1/test0 /mnt0
```

- 2 Create four data Storage Checkpoints on this file system, note the order of creation, and list them:

```
# fsckptadm create oldest /mnt0
# fsckptadm create older /mnt0
# fsckptadm create old /mnt0
# fsckptadm create latest /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime          = Mon 26 Jul 11:56:55 2004
  mtime          = Mon 26 Jul 11:56:55 2004
  flags          = largefiles
old:
  ctime          = Mon 26 Jul 11:56:51 2004
  mtime          = Mon 26 Jul 11:56:51 2004
  flags          = largefiles
older:
  ctime          = Mon 26 Jul 11:56:46 2004
  mtime          = Mon 26 Jul 11:56:46 2004
  flags          = largefiles
oldest:
  ctime          = Mon 26 Jul 11:56:41 2004
  mtime          = Mon 26 Jul 11:56:41 2004
  flags          = largefiles
```

- 3 Try to convert synchronously the latest Storage Checkpoint to a nodata Storage Checkpoint. The attempt will fail because the Storage Checkpoints older than the latest Storage Checkpoint are data Storage Checkpoints, namely the Storage Checkpoints old, older, and oldest:

```
# fsckptadm -s set nodata latest /mnt0
UX:vxfs fsckptadm: ERROR: V-3-24632: Storage Checkpoint
set failed on latest. File exists (17)
```

- 4 You can instead convert the `latest` Storage Checkpoint to a `nodata` Storage Checkpoint in a delayed or asynchronous manner.

```
# fsckptadm set nodata latest /mnt0
```

- 5 List the Storage Checkpoints, as in the following example. You will see that the `latest` Storage Checkpoint is marked for conversion in the future.

```
# fsckptadm list /mnt0
/mnt0
latest:
  ctime           = Mon 26 Jul 11:56:55 2004
  mtime           = Mon 26 Jul 11:56:55
  flags           = nodata, largefiles, delayed
old:
  ctime           = Mon 26 Jul 11:56:51 2004
  mtime           = Mon 26 Jul 11:56:51 2004
  flags           = largefiles
older:
  ctime           = Mon 26 Jul 11:56:46 2004
  mtime           = Mon 26 Jul 11:56:46 2004
  flags           = largefiles
oldest:
  ctime           = Mon 26 Jul 11:56:41 2004
  mtime           = Mon 26 Jul 11:56:41 2004
  flags           = largefiles
```

Creating a delayed `nodata` Storage Checkpoint

You can combine the three previous steps and create the `latest` Storage Checkpoint as a `nodata` Storage Checkpoint. The creation process will detect the presence of the older data Storage Checkpoints and create the `latest` Storage Checkpoint as a delayed `nodata` Storage Checkpoint.

To create a delayed nodata Storage Checkpoint

1 Remove the latest Storage Checkpoint.

```
# fsckptadm remove latest /mnt0
# fsckptadm list /mnt0
/mnt0
old:
  ctime          = Mon 26 Jul 11:56:51 2004
  mtime          = Mon 26 Jul 11:56:51 2004
  flags          = largefiles
older:
  ctime          = Mon 26 Jul 11:56:46 2004
  mtime          = Mon 26 Jul 11:56:46 2004
  flags          = largefiles
oldest:
  ctime          = Mon 26 Jul 11:56:41 2004
  mtime          = Mon 26 Jul 11:56:41 2004
  flags          = largefiles
```

2 Recreate the latest Storage Checkpoint as a nodata Storage Checkpoint.

```
# fsckptadm -n create latest /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime          = Mon 26 Jul 12:06:42 2004
  mtime          = Mon 26 Jul 12:06:42 2004
  flags          = nodata, largefiles, delayed
old:
  ctime          = Mon 26 Jul 11:56:51 2004
  mtime          = Mon 26 Jul 11:56:51 2004
  flags          = largefiles
older:
  ctime          = Mon 26 Jul 11:56:46 2004
  mtime          = Mon 26 Jul 11:56:46 2004
  flags          = largefiles
oldest:
  ctime          = Mon 26 Jul 11:56:41 2004
  mtime          = Mon 26 Jul 11:56:41 2004
  flags          = largefiles
```

3 Convert the `oldest` Storage Checkpoint to a `nodata` Storage Checkpoint because no older Storage Checkpoints exist that contain data in the file system.

Note: This step can be done synchronously.

```
# fsckptadm -s set nodata oldest /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime           = Mon 26 Jul 12:06:42 2004
  mtime           = Mon 26 Jul 12:06:42 2004
  flags           = nodata, largefiles, delayed
old:
  ctime           = Mon 26 Jul 11:56:51 2004
  mtime           = Mon 26 Jul 11:56:51 2004
  flags           = largefiles
older:
  ctime           = Mon 26 Jul 11:56:46 2004
  mtime           = Mon 26 Jul 11:56:46 2004
  flags           = largefiles
oldest:
  ctime           = Mon 26 Jul 11:56:41 2004
  mtime           = Mon 26 Jul 11:56:41 2004
  flags           = nodata, largefiles
```

4 Remove the `older` and `old` Storage Checkpoints.

```
# fsckptadm remove older /mnt0
# fsckptadm remove old /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime          = Mon 26 Jul 12:06:42 2004
  mtime          = Mon 26 Jul 12:06:42 2004
  flags          = nodata, largefiles
oldest:
  ctime          = Mon 26 Jul 11:56:41 2004
  mtime          = Mon 26 Jul 11:56:41 2004
  flags          = nodata, largefiles
```

Note: After you remove the `older` and `old` Storage Checkpoints, the `latest` Storage Checkpoint is automatically converted to a `nodata` Storage Checkpoint because the only remaining `older` Storage Checkpoint (`oldest`) is already a `nodata` Storage Checkpoint.

Storage Checkpoint space management considerations

Several operations, such as removing or overwriting a file, can fail when a file system containing Storage Checkpoints runs out of space. If the system cannot allocate sufficient space, the operation will fail.

Database applications usually preallocate storage for their files and may not expect a write operation to fail. If a file system runs out of space, the kernel automatically removes Storage Checkpoints and attempts to complete the write operation after sufficient space becomes available. The kernel removes Storage Checkpoints to prevent commands, such as `rm`, from failing under an out-of-space (`ENOSPC`) condition.

When the kernel automatically removes the Storage Checkpoints, it applies the following policies:

- Remove as few Storage Checkpoints as possible to complete the operation.
- Never select a non-removable Storage Checkpoint.
- Select a `nodata` Storage Checkpoint only when data Storage Checkpoints no longer exist.

- Remove the oldest Storage Checkpoint first.

Restoring from a Storage Checkpoint

Mountable data Storage Checkpoints on a consistent and undamaged file system can be used by backup and restore applications to restore either individual files or an entire file system. Restoration from Storage Checkpoints can also help recover incorrectly modified files, but typically cannot recover from hardware damage or other file system integrity problems.

Note: For hardware or other integrity problems, Storage Checkpoints must be supplemented by backups from other media.

Files can be restored by copying the entire file from a mounted Storage Checkpoint back to the primary fileset. To restore an entire file system, you can designate a mountable data Storage Checkpoint as the primary fileset using the `fsckpt_restore` command.

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

When using the `fsckpt_restore` command to restore a file system from a Storage Checkpoint, all changes made to that file system after that Storage Checkpoint's creation date are permanently lost. The only Storage Checkpoints and data preserved are those that were created at the same time, or before, the selected Storage Checkpoint's creation. The file system cannot be mounted at the time that `fsckpt_restore` is invoked.

Note: Individual files can also be restored very efficiently by applications using the `fsckpt_fbmap(3)` library function to restore only modified portions of a files data.

Restoring a file from a Storage Checkpoint

The following example restores a file, `MyFile.txt`, which resides in your home directory, from the Storage Checkpoint `CKPT1` to the device

`/dev/vx/dsk/dg1/vol-01`. The mount point for the device is `/home`.

To restore a file from a Storage Checkpoint

- 1 Create the Storage Checkpoint CKPT1 of /home.

```
$ fckptadm create CKPT1 /home
```

- 2 Mount Storage Checkpoint CKPT1 on the directory /home/checkpoints/mar_4.

```
$ mount -V vxfs -o ckpt=CKPT1 /dev/vx/dsk/dg1/vol- \
01:CKPT1 /home/checkpoints/mar_4
```

- 3 Delete the file MyFile.txt from your home directory.

```
$ cd /home/users/me
$ rm MyFile.txt
```

- 4 Go to the /home/checkpoints/mar_4/users/me directory, which contains the image of your home directory.

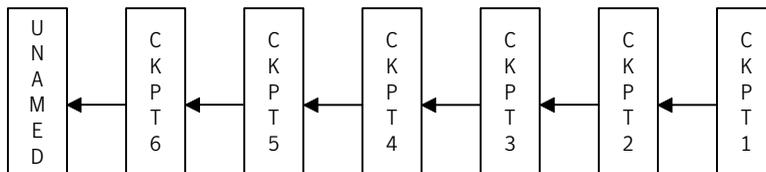
```
$ cd /home/checkpoints/mar_4/users/me
$ ls -l
-rw-r--r-- 1 me staff 14910 Mar 4 17:09 MyFile.txt
```

- 5 Copy the file MyFile.txt to your home directory.

```
$ cp MyFile.txt /home/users/me
$ cd /home/users/me
$ ls -l
-rw-r--r-- 1 me staff 14910 Mar 4 18:21 MyFile.txt
```

Restoring a file system from a Storage Checkpoint

The following example restores a file system from the Storage Checkpoint CKPT3. The filesets listed before the restoration show an unnamed root fileset and six Storage Checkpoints.



To restore a file system from a Storage Checkpoint

1 Run the `fsckpt_restore` command:

```
# fsckpt_restore -l /dev/vx/dsk/dg1/vol2
/dev/vx/dsk/dg1/vol2:
UNNAMED:
  ctime      = Thu 08 May 2004 06:28:26 PM PST
  mtime      = Thu 08 May 2004 06:28:26 PM PST
  flags      = largefiles, file system root
CKPT6:
  ctime      = Thu 08 May 2004 06:28:35 PM PST
  mtime      = Thu 08 May 2004 06:28:35 PM PST
  flags      = largefiles
CKPT5:
  ctime      = Thu 08 May 2004 06:28:34 PM PST
  mtime      = Thu 08 May 2004 06:28:34 PM PST
  flags      = largefiles, nomount
CKPT4:
  ctime      = Thu 08 May 2004 06:28:33 PM PST
  mtime      = Thu 08 May 2004 06:28:33 PM PST
  flags      = largefiles
CKPT3:
  ctime      = Thu 08 May 2004 06:28:36 PM PST
  mtime      = Thu 08 May 2004 06:28:36 PM PST
  flags      = largefiles
CKPT2:
  ctime      = Thu 08 May 2004 06:28:30 PM PST
  mtime      = Thu 08 May 2004 06:28:30 PM PST
  flags      = largefiles
CKPT1:
  ctime      = Thu 08 May 2004 06:28:29 PM PST
  mtime      = Thu 08 May 2004 06:28:29 PM PST
  flags      = nodata, largefiles
```

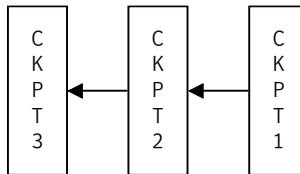
2 In this example, select the Storage Checkpoint CKPT3 as the new root fileset:

```
Select Storage Checkpoint for restore operation
or <Control/D> (EOF) to exit
or <Return> to list Storage Checkpoints: CKPT3
CKPT3:
  ctime          = Thu 08 May 2004 06:28:31 PM PST
  mtime          = Thu 08 May 2004 06:28:36 PM PST
  flags          = largefiles
UX:vxfs fscckpt_restore: WARNING: V-3-24640: Any file system
changes or Storage Checkpoints made after
Thu 08 May 2004 06:28:31 PM PST will be lost.
```

3 Type **y** to restore the file system from CKPT3:

```
Restore the file system from Storage Checkpoint CKPT3 ?
(ynq) y
(Yes)
UX:vxfs fsckpt_restore: INFO: V-3-23760: File system
restored from CKPT3
```

If the filesets are listed at this point, it shows that the former UNNAMED root fileset and CKPT6, CKPT5, and CKPT4 were removed, and that CKPT3 is now the primary fileset. CKPT3 is now the fileset that will be mounted by default.



4 Run the `fsckpt_restore` command:

```
# fsckpt_restore -l /dev/vx/dsk/dg1/vol2
/dev/vx/dsk/dg1/vol2:
CKPT3:
  ctime      = Thu 08 May 2004 06:28:31 PM PST
  mtime      = Thu 08 May 2004 06:28:36 PM PST
  flags      = largefiles, file system root
CKPT2:
  ctime      = Thu 08 May 2004 06:28:30 PM PST
  mtime      = Thu 08 May 2004 06:28:30 PM PST
  flags      = largefiles
CKPT1:
  ctime      = Thu 08 May 2004 06:28:29 PM PST
  mtime      = Thu 08 May 2004 06:28:29 PM PST
  flags      = nodata, largefiles
Select Storage Checkpoint for restore operation
or <Control/D> (EOF) to exit
or <Return> to list Storage Checkpoints:
```

Storage Checkpoint quotas

VxFS provides options to the `fscckptadm` command interface to administer Storage Checkpoint quotas. Storage Checkpoint quotas set the following limits on the amount of space used by all Storage Checkpoints of a primary file set:

hard limit	An absolute limit that cannot be exceeded. If a hard limit is exceeded, all further allocations on any of the Storage Checkpoints fail, but existing Storage Checkpoints are preserved.
soft limit	Must be lower than the hard limit. If a soft limit is exceeded, no new Storage Checkpoints can be created. The number of blocks used must return below the soft limit before more Storage Checkpoints can be created. An alert and console message are generated.

In case of a hard limit violation, various solutions are possible, enacted by specifying or not specifying the `-f` option for the `fscckptadm` utility.

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

Specifying or not specifying the `-f` option has the following effects:

- If the `-f` option is not specified, one or many removable Storage Checkpoints are deleted to make space for the operation to succeed. This is the default solution.
- If the `-f` option is specified, all further allocations on any of the Storage Checkpoints fail, but existing Storage Checkpoints are preserved.

Note: Sometimes if a file is removed while it is opened by another process, the removal process is deferred until the last close. Because the removal of a file may trigger pushing data to a “downstream” Storage Checkpoint (that is, the next older Storage Checkpoint), a fileset hard limit quota violation may occur. In this scenario, the hard limit is relaxed to prevent an inode from being marked bad. This is also true for some asynchronous inode operations.

Administering FileSnaps

This chapter includes the following topics:

- [About FileSnaps](#)
- [Properties of FileSnaps](#)
- [Creating FileSnaps](#)
- [Concurrent I/O to FileSnaps](#)
- [Copy-on-write and FileSnaps](#)
- [Reading from FileSnaps](#)
- [Block map fragmentation and FileSnaps](#)
- [Backup and FileSnaps](#)
- [Using FileSnaps](#)
- [Best practices with FileSnaps](#)
- [Comparison of the logical size output of the `fsadm -S shared`, `du`, and `df` commands](#)

About FileSnaps

A FileSnap is an atomic space-optimized copy of a file in the same name space, stored in the same file system. Veritas File System (VxFS) supports snapshots of files in the VxFS 5.1 SP1 release and later, and on file system disk layout Version 8 and later.

The `vxfilesnap` command creates a snapshot of a file, and the basic command syntax is similar to the `cp` command. If the source file argument is a directory, the `vxfilesnap` command creates a snapshot of all of the regular files in the first

level of the directory. The `vxfilesnap` command preserves the inode identify of the destination file, if the destination file exists.

All regular file operations are supported on the FileSnap, and VxFS does not distinguish the FileSnap in any way.

See the `vxfilesnap(1)` manual page.

Properties of FileSnaps

FileSnaps provide an ability to snapshot objects that are smaller in granularity than a file system or a volume. The ability to snapshot parts of a file system name space is required for application-based or user-based management of data stored in a file system. This is useful when a file system is shared by a set of users or applications or the data is classified into different levels of importance in the same file system.

FileSnaps provide non-root users the ability to snapshot data that they own, without requiring administrator privileges. This enables users and applications to version, backup, and restore their data by scheduling snapshots at appropriate points of their application cycle. Restoring from a FileSnap is as simple as specifying a snapshot as the source file and the original file as the destination file as the arguments for the `vxfilesnap` command.

FileSnap creation locks the source file as read-only and locks the destination file exclusively for the duration of the operation, thus creating the snapshots atomically. The rest of the files in the file system can be accessed with no I/O pause while FileSnap creation is in progress. Read access to the source file is also uninterrupted while the snapshot creation is in progress. This allows for true sharing of a file system by multiple users and applications in a non-intrusive fashion.

The name space relationship between source file and destination file is defined by the user-issued `vxfilesnap` command by specifying the destination file path. Veritas File System (VxFS) neither differentiates between the source file and the destination file, nor does it maintain any internal relationships between these two files. Once the snapshot is completed, the only shared property between the source file and destination file are the data blocks and block map shared by them.

The number of FileSnaps of a file is practically unlimited. The technical limit is the maximum number of files supported by the VxFS file system, which is one billion files per file set. When thousands of FileSnaps are created from the same file and each of these snapshot files is simultaneously read and written to by thousands of threads, FileSnaps scale very well due to the design that results in no contention of the shared blocks when unsharing happens due to an overwrite. The performance seen for the case of unsharing shared blocks due to an overwrite

with FileSnaps is closer to that of an allocating write than that of a traditional copy-on-write.

In disk layout Version 8, to support block or extent sharing between the files, reference counts are tracked for each shared extent. VxFS processes reference count updates due to sharing and unsharing of extents in a delayed fashion. Also, an extent that is marked shared once will not go back to unshared until all the references are gone. This is to improve the FileSnap creation performance and performance of data extent unsharing. However, this in effect results in the shared block statistics for the file system to be only accurate to the point of the processing of delayed reclamation. In other words, the shared extent statistics on the file system and a file could be stale, depending on the state of the file system.

Creating FileSnaps

A single thread creating FileSnaps of the same file can create over ten thousand snapshots per minute. FileSnaps can be used for fast provisioning of new virtual machines by cloning a virtual machine golden image, where the golden image is stored as a file in a VxFS file system or Veritas Storage Foundation Cluster File System (SFCFS) file system, which is used as a data store for a virtual environment.

Concurrent I/O to FileSnaps

FileSnaps design and implementation ensures that concurrent reads or writes to different snapshots of the same file perform as if these were independent files. Even though the extents are shared between snapshots of the same file, the sharing has no negative impact on concurrent I/O.

Copy-on-write and FileSnaps

Veritas File System (VxFS) supports an option to do lazy copy-on-write when a region of a file referred to by a shared extent is overwritten. A typical copy-on-write implementation involves reading the old data, allocating a new block, copying or writing the old data to the new block synchronously, and writing the new data to the new block. This results in a worst case possibility of one or more allocating transactions, followed by a read, followed by a synchronous write and another write that conforms to the I/O behavior requested for the overwrite. This sequence makes typical copy-on-write a costly operation. The VxFS lazy copy-on-write implementation does not copy the old data to the newly allocated block and hence does not have to read the old data either, as long as the new data covers the entire block. This behavior combined with delayed processing of shared extent accounting makes the lazy copy-on-write complete in times comparable to that of an allocating

write. However, in the event of a server crash, when the server has not flushed the new data to the newly allocated blocks, the data seen on the overwritten region would be similar to what you would find in the case of an allocating write where the server has crashed before the data is flushed. This is not the default behavior and with the default behavior the data that you find in the overwritten region will be either the new data or the old data.

Reading from FileSnaps

For regular read requests, Veritas File System (VxFS) only caches a single copy of a data page in the page cache for a given shared data block, even though the shared data block could be accessed from any of the FileSnaps or the source file. Once the shared data page is cached, any subsequent requests via any of the FileSnaps or the source file is serviced from the page cache. This eliminates duplicate read requests to the disk, which results in lower I/O load on the array. This also reduces the page cache duplication, which results in efficient usage of system page cache with very little cache churning when thousands of FileSnaps are accessed.

Block map fragmentation and FileSnaps

The block map of the source file is shared by the snapshot file. When data is overwritten on a previously shared region, the block map of the file to which the write happens gets changed. In cases where the shared data extent of a source file is larger than the size of the overwrite request to the same region, the block map of the file that is written to becomes more fragmented.

Backup and FileSnaps

A full backup of a VxFS file system that has shared blocks may require as much space in the target as the number of total logical references to the physical blocks in the source file system.

For example, if you have a 20 GB file from which one thousand FileSnaps were created, the total number of logical block references is approximately 20 TB. While the VxFS file system only requires a little over 20 GB of physical blocks to store the file and the file's one thousand snapshots, the file system requires over 20 TB of space on the backup target to back up the file system, assuming the backup target does not have deduplication support.

Using FileSnaps

[Table 20-1](#) provides a list of Veritas File System (VxFS) commands that enable you to administer FileSnaps.

Table 20-1

Command	Functionality
<code>fiostat</code>	The <code>fiostat</code> command has the <code>-S</code> option to display statistics for each interval. Otherwise, the command displays the accumulated statistics for the entire time interval.
<code>fsadm</code>	The <code>fsadm</code> command has the <code>-S</code> option to report shared block usage in the file system. You can use this option to find out the storage savings achieved through FileSnaps and how much real storage is required if all of the files are full copies. See the <code>fsadm_vxfs(1M)</code> manual page.
<code>fsmap</code>	The <code>fsmap</code> command has the <code>-c</code> option to report the count of the total number of physical blocks consumed by a file, and how many of those blocks might not be private to a given file. See the <code>fsmap(1)</code> manual page.
<code>mkfs</code>	Use the <code>mkfs</code> command to make a disk layout Version 8 file system by specifying <code>-o version=8</code> . VxFS internally maintains a list of delayed operations on shared extent references and the size of this list (<code>rcqsize</code>) defaults to a value that is a function of the file system size, but can be changed when the file system is made. See the <code>mkfs_vxfs(1M)</code> manual page.
<code>vxfilesnap</code>	Use the <code>vxfilesnap</code> command to create a snapshot of a file or set of files or files in a directory. You can also use the <code>vxfilesnap</code> command to restore a older version of the file to the current file. See the <code>vxfilesnap(1)</code> manual page.
<code>vxtunefs</code>	The <code>vxtunefs</code> command supports an option to enable lazy copy-on-write tuneable, <code>lazy_copyonwrite</code> , on the file system, for better performance. See the <code>vxtunefs(1M)</code> manual page.

Best practices with FileSnaps

The key to obtaining maximum performance with FileSnaps is to minimize the copy-on-write overhead. You can achieved this by enabling lazy copy-on-write. Lazy copy-on-write is easy to enable and usually results in significantly better performance. If lazy copy-on-write is not a viable option for the use case under consideration, an efficient allocation of the source file can reduce the need of copy-on-write.

Virtual desktops

Virtual desktop infrastructure (VDI) operating system boot images are a very good use case for FileSnaps. The parts of the boot images that may change are user profile, page files (or swap for UNIX/Linux) and application data. It would be ideal to separate such data from boot images to minimize unsharing. It is also important to allocate a single extent to the master boot image file.

Virtual desktop example

The following example uses a 4 GB master boot image that has a single extent that will be shared by all snapshots.

```
# touch /vdi_images/master_image  
# /opt/VRTS/bin/setext -r 4g -f chgsize /vdi_images/master_image
```

The `master_image` file can be presented as a disk device to the virtual machine for installing operating system. Once the operating system is installed and configured, the file is ready for snapshots.

Write intensive applications

When virtual machines are spawned to perform certain tasks that are write intensive, significant amount of unsharing can take place. Symantec recommends that you optimize performance by enabling lazy copy-on-write. If the use case does not allow enabling lazy copy-on-write, with careful planning, you can reduce the occurrence of unsharing. The easiest way to reduce unsharing is to separate the application data to a file other than the boot image. If this is not possible due to the nature of your applications, then you can take actions similar to the following example.

Write intensive applications example

Assume that the disk space required for a boot image and the application data is 20 GB. Out of this, only 4 GB is used by the operating system and the remaining

16 GB is the space for applications to write. Any data or binaries that are required by each instance of the virtual machine can still be part of the first 4 GB of the shared extent. Since most of the writes are expected to take place on the 16 GB portion, you should allocate the master image in such a way that the 16 GB of space is not shared, as shown in the following commands:

```
# touch /vdi_images/master_image
# /opt/VRTS/bin/setext -r 4g -f chgsize /vdi_images/master_image
# dd if=/dev/zero of=/vdi_images/master_image seek=16777215 \
bs=1024 count=1
```

The last command creates a 16 GB hole at the end of the file. Since holes do not have any extents allocated, the writes to hole do not need to be unshared.

Data mining, reporting, and testing

It is very common to create one or more copies of production data for the purpose of generating reports, mining, and testing. These cases frequently update the copies of the data with the most current data and one or more copies of the data always exists. FileSnaps can be used to create multiple copies instantly. The application that uses the original data may see a slight performance hit due to the unsharing of data that may take place during updates. This slight impact on performance may still be present even when all FileSnaps have been deleted. An important point to note though is that many of these use cases almost always have one or more copies all of the time.

Comparison of the logical size output of the `fsadm -S shared`, `du`, and `df` commands

The `fsadm -S shared`, `du`, and `df` commands report different values for the size of a FileSnap. The `fsadm -S shared` command displays this size as the "logical size," which is the logical space consumed, in kilobytes, and accounts for both exclusive blocks and shared blocks. This value represents the actual disk space needed if the file system did not have any shared blocks. The value from the `fsadm -S shared` command differs from the output of `du -sk` command since the `du` command does not track the blocks consumed by VxFS structural files. As a result, the output of the `du -sk` command is less than the logical size output reported by the `fsadm -S shared` command.

The following examples show output from the `fsadm -S shared`, `du`, and `df` commands:

Comparison of the logical size output of the fsadm -S shared, du, and df commands

```

# mkfs -V vxfs -o version=8 /dev/vx/rdisk/dg/vol3
version 8 layout
104857600 sectors, 52428800 blocks of size 1024, log size 65536 blocks
rcq size 4096 blocks
largefiles supported

# mount -V vxfs /dev/vx/dsk/dg/vol3 /mnt

# df -k /mnt
Filesystem          1K-blocks    Used   Available Use% Mounted on
/dev/vx/dsk/dg1/vol3 52428800    83590  49073642  1% /mnt

# /opt/VRTS/bin/fsadm -S shared /mnt
Mountpoint    Size (KB)    Available (KB)    Used (KB)    Logical_Size (KB)    Shared
/mnt          52428800    49073642          83590        83590                0%

# du -sk /mnt
0           /mnt

# dd if=/dev/zero of=/mnt/foo bs=1024 count=10
10+0 records in
10+0 records out
10240 bytes (10 kB) copied, 0.018901 seconds, 542 kB/s

# vxfilesnap /mnt/foo /mnt/foo.snap

# df -k /mnt
Filesystem          1K-blocks    Used   Available Use% Mounted on
/dev/vx/dsk/dg1/vol3 52428800    83600  49073632  1% /mnt

# /opt/VRTS/bin/fsadm -S shared /mnt
Mountpoint    Size (KB)    Available (KB)    Used (KB)    Logical_Size (KB)    Shared
/mnt          52428800    49073632          83600        83610                0%

# du -sk /mnt
20          /mnt

```

Backing up and restoring with Netbackup in an SFHA environment

This chapter includes the following topics:

- [About Veritas NetBackup](#)
- [About using Veritas NetBackup for backup and restore for DB2](#)
- [About using NetBackup for backup and restore for Sybase](#)
- [About using Veritas NetBackup to backup and restore Quick I/O files for DB2](#)
- [About using Veritas NetBackup to backup and restore Quick I/O files for Sybase](#)

About Veritas NetBackup

Veritas NetBackup provides backup, archive, and restore capabilities for database files and directories contained on client systems in a client-server network. NetBackup server software resides on platforms that manage physical backup storage devices. The NetBackup server provides robotic control, media management, error handling, scheduling, and a repository of all client backup images.

Administrators can set up schedules for automatic, unattended full and incremental backups. These backups are managed entirely by the NetBackup server. The administrator can also manually back up clients. Client users can perform backups, archives, and restores from their client system, and once started, these operations also run under the control of the NetBackup server.

Veritas NetBackup can be configured for DB2 in an Extended Edition (EE) or Extended-Enterprise Edition (EEE) environment. For detailed information and instructions on configuring DB2 for EEE, see “Configuring for a DB2 EEE (DPF) Environment” in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

Veritas NetBackup, while not a shipped component of Veritas Storage Foundation Enterprise products, can be purchased separately.

About using Veritas NetBackup for backup and restore for DB2

With Veritas NetBackup, you can perform high performance, online (hot) backups of databases that must be available on a 24x7 basis. NetBackup supports the Extended Edition (EE) and the Enterprise Extended Edition (EEE) environments. NetBackup also supports DPF for DB2 8.1 and higher.

Veritas NetBackup lets you back up and restore database files and directories. You can set up schedules for automatic, unattended database backup, as well as full or incremental backup. These backups are managed entirely by the NetBackup server. You can also manually back up database files from any of the NetBackup clients. Client users can perform database backups and restores from their client systems on demand. The topics below summarize NetBackup's backup and restore capabilities for DB2. More information on backup and restore procedures and supported EE and EEE versions is available in the system administrator's guide.

See 'Using NetBackup for DB2 on UNIX' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

Components of Veritas NetBackup

Veritas NetBackup for DB2 has the following features:

- Media and device management
- Scheduling facilities
- Multiplexed backups and restores
- Transparent execution of both DB2 and regular file system backup and restore operations
- Shared devices and tapes used during other file backups
- Centralized and networked backup operations
- Parallel backup and restore operations

- Incremental backups of DB2 databases

About performing a backup

There are two types of DB2 backups: database logs and archive logs. These two types of backups can be performed by NetBackup automatically, manually, or by using the `DB2 BACKUP DATABASE` command. More information on performing a backup is available in the system administrator's guide.

See 'Performing a Backup' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

About DB2 policy backups

You can back up a DB2 policy automatically or manually. The most convenient way to back up your database is to set up schedules for automatic backups.

About DB2 restore

The procedure for restoring a DB2 database depends on the database involved and the problems that you have on your system. You can browse the backups using the `db2 list history` command or using the NetBackup `bplist` command before restoring. More information and a complete description of how to recover a DB2 database is available in the system administrator's guide.

See the *DB2 UDB Administration Guide Data Recovery and High Availability Guide*.

About configuring Veritas NetBackup for a DB2 EEE (DPF) environment

Veritas NetBackup can be configured for DB2 in an Extended Edition (EE), Extended-Enterprise Edition (EEE), or Database Partitioning Feature (DPF) environment. Two types of DB2 backup policies are required. One is used to backup the catalog nodes and the other is used to backup all the nodes, including the catalog node. Detailed information and instructions on configuring DB2 for EEE is available in the system administrator's guide.

See 'Configuration for a DB2 EEE (DPF) Environment' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

About using NetBackup for backup and restore for Sybase

Veritas NetBackup for Sybase is not included in the standard Veritas Database Edition. The information included here is for reference only.

Veritas NetBackup for Sybase integrates the database backup and recovery capabilities of Sybase Backup Server with the backup and recovery management capabilities of NetBackup.

Veritas NetBackup works with Sybase APIs to provide high-performance backup and restore for Sybase dataservers. With Veritas NetBackup, you can set up schedules for automatic, unattended backups for Sybase ASE dataservers (NetBackup clients) across the network. These backups can be full database dumps or incremental backups (transaction logs) and are managed by the NetBackup server. You can also manually backup dataservers. The Sybase `dump` and `load` commands are used to perform backups and restores.

Veritas NetBackup has both graphical and menu driven user interfaces to suit your needs.

For details, refer to *Veritas NetBackup System Administrator's Guide for UNIX*.

About using Veritas NetBackup to backup and restore Quick I/O files for DB2

Veritas NetBackup does not follow symbolic links when backing up files. Typical backup management applications are designed this way to avoid backing up the same data twice. This would happen if both the link and the file it points to were included in the list of files to be backed up.

A Quick I/O file consists of two components: a hidden file with the space allocated for it, and a link that points to the Quick I/O interface of the hidden file. Because NetBackup does not follow symbolic links, you must specify both the Quick I/O link and its hidden file in the list of files to be backed up.

To view all files and their attributes in the `db01` directory:

```
$ ls -la /db01

total 2192

drwxr-xr-x  2 root  root  96  Oct 20 17:39  .
drwxr-xr-x  9 root  root 8192 Oct 20 17:39  ..
-rw-r--r--  1 db2  dba 1048576 Oct 20 17:39  .dbfile
lrwxrwxrwx  1 db2  dba  22  Oct 20 17:39  dbfile ->\
.dbfile::cdev:vxfs:
```

In the example above, you must include both the symbolic link `dbfile` and the hidden file `.dbfile` in the file list of the backup class.

If you want to back up all Quick I/O files in a directory, you can simplify the process by just specifying the directory to be backed up. In this case, both components of each Quick I/O file will be properly backed up. In general, you should specify directories to be backed up unless you only want to back up some, but not all files, in those directories.

Because NetBackup is tightly integrated with the Veritas Database Edition for DB2, NetBackup backs up extent attributes of a Quick I/O file and restores them accordingly. Quick I/O files can then be backed up and restored as regular files using NetBackup, while preserving the Quick I/O file's extent reservation. Without this feature, restoring the file could cause the loss of contiguous reservation, which can degrade performance.

When restoring a Quick I/O file, if both the symbolic link and the hidden file already exist, NetBackup will restore both components from the backup image. If either one of or both of the two components are missing, NetBackup creates or overwrites as needed.

About using Veritas NetBackup to backup and restore Quick I/O files for Sybase

Veritas NetBackup does not follow symbolic links when backing up files. Typical backup management applications are designed this way to avoid backing up the same data twice. This would happen if both the link and the file it points to were included in the list of files to be backed up.

A Quick I/O file consists of two components: a hidden file with the space allocated for it, and a link that points to the Quick I/O interface of the hidden file. Because NetBackup does not follow symbolic links, you must specify both the Quick I/O link and its hidden file in the list of files to be backed up.

To view all files and their attributes in the db01 directory:

```
$ ls -la /db01

total 2192

drwxr-xr-x  2 root  root  96  Oct 20 17:39  .
drwxr-xr-x  9 root  root 8192 Oct 20 17:39  ..
-rw-r--r--  1 db2  dba 1048576 Oct 20 17:39  .dbfile
lrwxrwxrwx  1 db2  dba  22  Oct 20 17:39  dbfile ->\
.dbfile::cdev:vxfs:
```

In the example above, you must include both the symbolic link `dbfile` and the hidden file `.dbfile` in the file list of the backup class.

If you want to back up all Quick I/O files in a directory, you can simplify the process by just specifying the directory to be backed up. In this case, both components of each Quick I/O file will be properly backed up. In general, you should specify directories to be backed up unless you only want to back up some, but not all files, in those directories.

When restoring a Quick I/O file, if both the symbolic link and the hidden file already exist, NetBackup will restore both components from the backup image. If either one of or both of the two components are missing, NetBackup creates or overwrites as needed.

Maximizing storage utilization

- [Chapter 22. Understanding storage tiering with SmartTier](#)
- [Chapter 23. Creating and administering volume sets](#)
- [Chapter 24. Multi-volume file systems](#)
- [Chapter 25. Administering SmartTier](#)

Understanding storage tiering with SmartTier

This chapter includes the following topics:

- [About SmartTier](#)
- [SmartTier building blocks](#)
- [How SmartTier works](#)
- [SmartTier in a High Availability \(HA\) environment](#)

About SmartTier

Note: SmartTier is the expanded and renamed feature previously known as Dynamic Storage Tiering (DST).

SmartTier matches data storage with data usage requirements. After data matching, the data can then be relocated based upon data usage and other requirements determined by the storage or database administrator (DBA).

As more and more data is retained over a period of time, eventually, some of that data is needed less frequently. The data that is needed less frequently still requires a large amount of disk space. SmartTier enables the database administrator to manage data so that less frequently used data can be moved to slower, less expensive disks. This also permits the frequently accessed data to be stored on faster disks for quicker retrieval.

Tiered storage is the assignment of different types of data to different storage types to improve performance and reduce costs. With SmartTier, storage classes

are used to designate which disks make up a particular tier. There are two common ways of defining storage classes:

- Performance, or storage, cost class: The most-used class consists of fast, expensive disks. When data is no longer needed on a regular basis, the data can be moved to a different class that is made up of slower, less expensive disks.
- Resilience class: Each class consists of non-mirrored volumes, mirrored volumes, and n-way mirrored volumes.

For example, a database is usually made up of data, an index, and logs. The data could be set up with a three-way mirror because data is critical. The index could be set up with a two-way mirror because the index is important, but can be recreated. The redo and archive logs are not required on a daily basis but are vital to database recovery and should also be mirrored.

SmartTier policies control initial file location and the circumstances under which existing files are relocated. These policies cause the files to which they apply to be created and extended on specific subsets of a file systems's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet specified naming, timing, access rate, and storage capacity-related conditions.

In addition to preset policies, you can manually move files to faster or slower storage with SmartTier, when necessary. You can also run reports that list active policies, display file activity, display volume usage, or show file statistics.

SmartTier building blocks

To use SmartTier, your storage must be managed using the following features:

- VxFS multi-volume file system
- VxVM volume set
- Volume tags
- SmartTier management at the file level
- SmartTier management at the sub-file level

About VxFS multi-volume file systems

Multi-volume file systems are file systems that occupy two or more virtual volumes. The collection of volumes is known as a volume set, and is made up of disks or disk array LUNs belonging to a single Veritas Volume Manager (VxVM) disk group. A multi-volume file system presents a single name space, making the existence

of multiple volumes transparent to users and applications. Each volume retains a separate identity for administrative purposes, making it possible to control the locations to which individual files are directed.

See [“About multi-volume support”](#) on page 323.

This feature is available only on file systems meeting the following requirements:

- The minimum disk group version is 140.
- The minimum file system layout version is 7 for file level SmartTier.
- The minimum file system layout version is 8 for sub-file level SmartTier.

To convert your existing VxFS system to a VxFS multi-volume file system, you must convert a single volume to a volume set.

See [“Converting a single volume file system to a multi-volume file system”](#) on page 327.

The VxFS volume administration utility (fsvoladm utility) can be used to administer VxFS volumes. The fsvoladm utility performs administrative tasks, such as adding, removing, resizing, encapsulating volumes, and setting, clearing, or querying flags on volumes in a specified Veritas File System.

See the fsvoladm (1M) manual page for additional information about using this utility.

About VxVM volume sets

Volume sets allow several volumes to be represented by a single logical object. Volume sets cannot be empty. All I/O from and to the underlying volumes is directed via the I/O interfaces of the volume set. The volume set feature supports the multi-volume enhancement to Veritas File System (VxFS). This feature allows file systems to make best use of the different performance and availability characteristics of the underlying volumes. For example, file system metadata could be stored on volumes with higher redundancy, and user data on volumes with better performance.

See the *Veritas Volume Manager Administrator's Guide*.

About volume tags

You make a VxVM volume part of a placement class by associating a volume tag with it. For file placement purposes, VxFS treats all of the volumes in a placement class as equivalent, and balances space allocation across them. A volume may have more than one tag associated with it. If a volume has multiple tags, the volume belongs to multiple placement classes and is subject to allocation and relocation policies that relate to any of the placement classes.

Warning: Multiple tagging should be used carefully.

A placement class is a SmartTier attribute of a given volume in a volume set of a multi-volume file system. This attribute is a character string, and is known as a volume tag.

See the *Veritas Volume Manager Administrator's Guide*.

How SmartTier works

SmartTier is a VxFS feature that enables you to allocate file storage space from different storage tiers according to rules you create. SmartTier provides a more flexible alternative compared to current approaches for tiered storage. Static storage tiering involves a manual one-time assignment of application files to a storage class, which is inflexible over a long term. Hierarchical Storage Management solutions typically require files to be migrated back into a file system name space before an application access request can be fulfilled, leading to latency and run-time overhead. In contrast, SmartTier allows organizations to:

- Optimize storage assets by dynamically moving a file to its optimal storage tier as the value of the file changes over time
- Automate the movement of data between storage tiers without changing the way users or applications access the files
- Migrate data automatically based on policies set up by administrators, eliminating operational requirements for tiered storage and downtime commonly associated with data movement

SmartTier leverages two key technologies included with Veritas Storage Foundation: support for multi-volume file systems and automatic policy-based placement of files within the storage managed by a file system. A multi-volume file system occupies two or more virtual storage volumes and thereby enables a single file system to span across multiple, possibly heterogeneous, physical storage devices. For example the first volume could reside on EMC Symmetrix DMX spindles, and the second volume could reside on EMC CLARiiON spindles. By presenting a single name space, multi-volumes are transparent to users and applications. This multi-volume file system remains aware of each volume's identity, making it possible to control the locations at which individual files are stored. When combined with the automatic policy-based placement of files, the multi-volume file system provides an ideal storage tiering facility, which moves data automatically without any downtime requirements for applications and users alike.

In a database environment, the access age rule can be applied to some files. However, some data files, for instance are updated every time they are accessed and hence access age rules cannot be used. SmartTier provides mechanisms to relocate portions of files as well as entire files to a secondary tier.

Moving files

SmartTier enables administrators of multi-volume VxFS file systems to manage the placement of files on individual volumes in a volume set by defining placement policies that control both initial file location and the circumstances under which existing files are relocated. These placement policies cause the files to which they apply to be created and extended on specific subsets of a file system's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet the specified naming, timing, access rate, and storage capacity-related conditions.

File-based movement:

- The administrator can create a file allocation policy based on filename extension before new files are created, which will create the datafiles on the appropriate tier during database creation.
- The administrator can also create a file relocation policy for database files or any types of files, which would relocate files based on how frequently a file is used.

Moving sub-file objects

SmartTier enables administrators of multi-volume VxFS file systems to manage the placement of file objects as well as entire files on individual volumes.

Using sub-file based movement you can:

- Move a set of ranges of a specified set of files of a specified set of mounts to a desired set of tiers on command.
- Move segments of files using automation to:
 - Monitor a set of files for collecting I/O statistics
 - Periodically collect and persist the statistics, cluster-wide if applicable
 - Periodically enforce the ranges of the registered sets of files based on their relative frequency of access to a desired set of tiers
 - Track the historical movements of those ranges

SmartTier in a High Availability (HA) environment

Veritas Cluster Server does not provide a bundled agent for volume sets. If issues arise with volumes or volume sets, the issues can only be detected at the DiskGroup and Mount resource levels.

The DiskGroup agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) disk group. This agent uses VxVM commands. When the value of the StartVolumes and StopVolumes attributes are both 1, the DiskGroup agent onlines and offlines the volumes during the import and deport operations of the disk group. When using volume sets, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains the volume are set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

The Mount agent brings online, takes offline, and monitors a file system or NFS client mount point.

If you are using any of the SmartTier for Oracle commands in a high availability (HA) environment, the time on each system in the cluster must be synchronized. Otherwise, the scheduled task may not be executed at the expected time after a service group failover.

Creating and administering volume sets

This chapter includes the following topics:

- [About volume sets](#)
- [Creating a volume set](#)
- [Adding a volume to a volume set](#)
- [Removing a volume from a volume set](#)
- [Listing details of volume sets](#)
- [Stopping and starting volume sets](#)
- [Raw device node access to component volumes](#)

About volume sets

Veritas File System (VxFS) uses volume sets to implement its Multi-Volume Support and SmartTier features.

Veritas Volume Manager (VxVM) provides the `vxvset` command to create and administer volume sets.

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

Volume sets have the following limitations:

- A maximum of 2048 volumes can be configured in a volume set.
- Only a Veritas File System is supported on a volume set.

- The first volume (index 0) in a volume set must be larger than the sum of the total volume size divided by 4000, the size of the VxFS intent log, and 1MB. Volumes 258 MB or larger should always suffice.
- Raw I/O from and to a volume set is not supported.
- Raw I/O from and to the component volumes of a volume set is supported under certain conditions.
See [“Raw device node access to component volumes”](#) on page 319.
- Volume sets can be used in place of volumes with the following `vxsnap` operations on instant snapshots: `addmir`, `dis`, `make`, `prepare`, `reattach`, `refresh`, `restore`, `rmir`, `split`, `syncpause`, `syncresume`, `syncstart`, `syncstop`, `syncwait`, and `unprepare`. The third-mirror break-off usage model for full-sized instant snapshots is supported for volume sets provided that sufficient plexes exist for each volume in the volume set.
- A full-sized snapshot of a volume set must itself be a volume set with the same number of volumes and the same volume index numbers as the parent. The corresponding volumes in the parent and snapshot volume sets are also subject to the same restrictions as apply between standalone volumes and their snapshots.

Creating a volume set

To create a volume set for use by Veritas File System (VxFS), use the following command:

```
# vxvset [-g diskgroup] -t vxfs make volset  
      volume
```

Here *volset* is the name of the volume set, and *volume* is the name of the first volume in the volume set. The `-t vxfs` option creates the volume set configured for use by VxFS. You must create the volume before running the command. `vxvset` will not automatically create the volume.

For example, to create a volume set named `myvset` that contains the volume `vol1`, in the disk group `mydg`, you would use the following command:

```
# vxvset -g mydg -t vxfs make myvset vol1
```

Adding a volume to a volume set

Having created a volume set containing a single volume, you can use the following command to add further volumes to the volume set:

```
# vxvset [-g diskgroup] [-f] addvol volset  
      volume
```

For example, to add the volume `vol2`, to the volume set `myvset`, use the following command:

```
# vxvset -g mydg addvol myvset vol2
```

Warning: The `-f` (force) option must be specified if the volume being added, or any volume in the volume set, is either a snapshot or the parent of a snapshot. Using this option can potentially cause inconsistencies in a snapshot hierarchy if any of the volumes involved in the operation is already in a snapshot chain.

Removing a volume from a volume set

To remove a component volume from a volume set, use the following command:

```
# vxvset [-g diskgroup] [-f] rmvol volset  
      volume
```

For example, the following commands remove the volumes, `vol1` and `vol2`, from the volume set `myvset`:

```
# vxvset -g mydg rmvol myvset vol1  
# vxvset -g mydg rmvol myvset vol2
```

Removing the final volume deletes the volume set.

Warning: The `-f` (force) option must be specified if the volume being removed, or any volume in the volume set, is either a snapshot or the parent of a snapshot. Using this option can potentially cause inconsistencies in a snapshot hierarchy if any of the volumes involved in the operation is already in a snapshot chain.

Listing details of volume sets

To list the details of the component volumes of a volume set, use the following command:

```
# vxvset [-g diskgroup] list [volset]
```

If the name of a volume set is not specified, the command lists the details of all volume sets in a disk group, as shown in the following example:

```
# vxvset -g mydg list
```

NAME	GROUP	NVOLS	CONTEXT
set1	mydg	3	-
set2	mydg	2	-

To list the details of each volume in a volume set, specify the name of the volume set as an argument to the command:

```
# vxvset -g mydg list set1
```

VOLUME	INDEX	LENGTH	KSTATE	CONTEXT
vol1	0	12582912	ENABLED	-
vol2	1	12582912	ENABLED	-
vol3	2	12582912	ENABLED	-

The context field contains details of any string that the application has set up for the volume or volume set to tag its purpose.

Stopping and starting volume sets

Under some circumstances, you may need to stop and restart a volume set. For example, a volume within the set may have become detached, as shown here:

```
# vxvset -g mydg list set1
```

VOLUME	INDEX	LENGTH	KSTATE	CONTEXT
vol1	0	12582912	DETACHED	-
vol2	1	12582912	ENABLED	-
vol3	2	12582912	ENABLED	-

To stop and restart one or more volume sets, use the following commands:

```
# vxvset [-g diskgroup] stop volset ...
# vxvset [-g diskgroup] start volset ...
```

For the example given previously, the effect of running these commands on the component volumes is shown below:

```
# vxvset -g mydg stop set1
```

```
# vxvset -g mydg list set1
```

VOLUME	INDEX	LENGTH	KSTATE	CONTEXT
--------	-------	--------	--------	---------

```

vol1          0          12582912      DISABLED      -
vol2          1          12582912      DISABLED      -
vol3          2          12582912      DISABLED      -

# vxvset -g mydg start set1

# vxvset -g mydg list set1

VOLUME        INDEX          LENGTH         KSTATE         CONTEXT
vol1          0          12582912      ENABLED         -
vol2          1          12582912      ENABLED         -
vol3          2          12582912      ENABLED         -

```

Raw device node access to component volumes

To guard against accidental file system and data corruption, the device nodes of the component volumes are configured by default not to have raw and block entries in the `/dev/vx/rdisk/diskgroup` and `/dev/vx/dsk/diskgroup` directories. As a result, applications are prevented from directly reading from or writing to the component volumes of a volume set.

If some applications, such as the raw volume backup and restore feature of the Symantec NetBackup™ software, need to read from or write to the component volumes by accessing raw device nodes in the `/dev/vx/rdisk/diskgroup` directory, this is supported by specifying additional command-line options to the `vxvset` command. Access to the block device nodes of the component volumes of a volume set is unsupported.

Warning: Writing directly to or reading from the raw device node of a component volume of a volume set should only be performed if it is known that the volume's data will not otherwise change during the period of access.

All of the raw device nodes for the component volumes of a volume set can be created or removed in a single operation. Raw device nodes for any volumes added to a volume set are created automatically as required, and inherit the access mode of the existing device nodes.

Access to the raw device nodes for the component volumes can be configured to be read-only or read-write. This mode is shared by all the raw device nodes for the component volumes of a volume set. The read-only access mode implies that any writes to the raw device will fail, however writes using the `ioctl` interface or by VxFS to update metadata are not prevented. The read-write access mode allows

direct writes via the raw device. The access mode to the raw device nodes of a volume set can be changed as required.

The presence of raw device nodes and their access mode is persistent across system reboots.

Note the following limitations of this feature:

- The disk group version must be 140 or greater.
- Access to the raw device nodes of the component volumes of a volume set is only supported for private disk groups; it is not supported for shared disk groups in a cluster.

Enabling raw device access when creating a volume set

To enable raw device access when creating a volume set, use the following form of the `vxvset make` command:

```
# vxvset [-g diskgroup] -o makedev=on \  
  [-o compvol_access={read-only|read-write}] \  
  [-o index] [-c "ch_addopt"] make vset  
  vol [index]
```

The `-o makedev=on` option enables the creation of raw device nodes for the component volumes at the same time that the volume set is created. The default setting is `off`.

If the `-o compvol_access=read-write` option is specified, direct writes are allowed to the raw device of each component volume. If the value is set to `read-only`, only reads are allowed from the raw device of each component volume.

If the `-o makedev=on` option is specified, but `-o compvol_access` is not specified, the default access mode is `read-only`.

If the `vxvset addvol` command is subsequently used to add a volume to a volume set, a new raw device node is created in `/dev/vx/rdisk/diskgroup` if the value of the `makedev` attribute is currently set to `on`. The access mode is determined by the current setting of the `compvol_access` attribute.

The following example creates a volume set, `myvset1`, containing the volume, `myvol1`, in the disk group, `mydg`, with raw device access enabled in read-write mode:

```
# vxvset -g mydg -o makedev=on -o compvol_access=read-write \  
  make myvset1 myvol1
```

Displaying the raw device access settings for a volume set

You can use the `vxprint -m` command to display the current settings for a volume set. If the `makedev` attribute is set to `on`, one of the following strings is displayed in the output:

```
vset_devinfo=on:read-only      Raw device nodes in read-only mode.
vset_devinfo=on:read-write    Raw device nodes in read-write mode.
```

A string is not displayed if `makedev` is set to `off`.

If the output from the `vxprint -m` command is fed to the `vxmake` command to recreate a volume set, the `vset_devinfo` attribute must set to `off`. Use the `vxvset set` command to re-enable raw device access with the desired access mode.

See [“Controlling raw device access for an existing volume set”](#) on page 321.

Controlling raw device access for an existing volume set

To enable or disable raw device node access for an existing volume set, use the following command:

```
# vxvset [-g diskgroup] [-f] set makedev={on|off} vset
```

The `makedev` attribute can be specified to the `vxvset set` command to create (`makedev=on`) or remove (`makedev=off`) the raw device nodes for the component volumes of a volume set. If any of the component volumes are open, the `-f` (force) option must be specified to set the attribute to `off`.

Specifying `makedev=off` removes the existing raw device nodes from the `/dev/vx/rdisk/diskgroup` directory.

If the `makedev` attribute is set to `off`, and you use the `mknod` command to create the raw device nodes, you cannot read from or write to those nodes unless you set the value of `makedev` to `on`.

The syntax for setting the `compvol_access` attribute on a volume set is:

```
# vxvset [-g diskgroup] [-f] set \
  compvol_access={read-only|read-write} vset
```

The `compvol_access` attribute can be specified to the `vxvset set` command to change the access mode to the component volumes of a volume set. If any of the component volumes are open, the `-f` (force) option must be specified to set the attribute to `read-only`.

The following example sets the `makedev=on` and `compvol_access=read-only` attributes on a volume set, `myvset2`, in the disk group, `mydg`:

```
# vxvset -g mydg set makedev=on myvset2
```

The next example sets the `compvol_access=read-write` attribute on the volume set, `myvset2`:

```
# vxvset -g mydg set compvol_access=read-write myvset2
```

The final example removes raw device node access for the volume set, `myvset2`:

```
# vxvset -g mydg set makedev=off myvset2
```

Multi-volume file systems

This chapter includes the following topics:

- [About multi-volume support](#)
- [About volume types](#)
- [Features implemented using multi-volume support](#)
- [Creating multi-volume file systems](#)
- [Converting a single volume file system to a multi-volume file system](#)
- [Adding a volume to and removing a volume from a multi-volume file system](#)
- [Volume encapsulation](#)
- [Reporting file extents](#)
- [Load balancing](#)
- [Converting a multi-volume file system to a single volume file system](#)

About multi-volume support

VxFS provides support for multi-volume file systems when used in conjunction with the Veritas Volume Manager. Using multi-volume support (MVS), a single file system can be created over multiple volumes, each volume having its own properties. For example, it is possible to place metadata on mirrored storage while placing file data on better-performing volume types such as RAID-1+0 (striped and mirrored). The volume must be in the same disk group as the volume set, and it cannot already be a member of another volume set.

The MVS feature also allows file systems to reside on different classes of devices, so that a file system can be supported from both inexpensive disks and from

expensive arrays. Using the MVS administrative interface, you can control which data goes on which volume types.

See the *Veritas Volume Manager Administrator's Guide*.

Note: Multi-volume support is available only on file systems using disk layout Version 6 or later.

About volume types

VxFS utilizes two types of volumes, one of which contains only data, referred to as `dataonly`, and the other of which can contain metadata or data, referred to as `metadataok`.

Data refers to direct extents, which contain user data, of regular files and named data streams in a file system.

Metadata refers to all extents that are not regular file or named data stream extents. This includes certain files that appear to be regular files, but are not, such as the File Change Log file.

A volume availability flag is set to specify if a volume is `dataonly` or `metadataok`. The volume availability flag can be set, cleared, and listed with the `fsvoladm` command.

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

Features implemented using multi-volume support

The following features can be implemented using multi-volume support:

- Controlling where files are stored can be selected at multiple levels so that specific files or file hierarchies can be assigned to different volumes. This functionality is available in the Veritas File System SmartTier feature. See “[About SmartTier](#)” on page 337.
- Placing the VxFS intent log on its own volume to minimize disk head movement and thereby increase performance.
- Separating Storage Checkpoints so that data allocated to a Storage Checkpoint is isolated from the rest of the file system.
- Separating metadata from file data.
- Encapsulating volumes so that a volume appears in the file system as a file. This is particularly useful for databases that are running on raw volumes.

- Guaranteeing that a `dataonly` volume being unavailable does not cause a `metadataok` volume to be unavailable.

To use the multi-volume file system features, Veritas Volume Manager must be installed and the volume set feature must be accessible. The volume set feature is separately licensed.

Volume availability

MVS guarantees that a `dataonly` volume being unavailable does not cause a `metadataok` volume to be unavailable. This allows you to mount a multi-volume file system even if one or more component `dataonly` volumes are missing.

The volumes are separated by whether metadata is allowed on the volume. An I/O error on a `dataonly` volume does not affect access to any other volumes. All VxFS operations that do not access the missing `dataonly` volume function normally.

Some VxFS operations that do not access the missing `dataonly` volume and function normally include the following:

- Mounting the multi-volume file system, regardless if the file system is read-only or read/write.
- Kernel operations.
- Performing a `fsck` replay. Logged writes are converted to normal writes if the corresponding volume is `dataonly`.
- Performing a full `fsck`.
- Using all other commands that do not access data on a missing volume.

Some operations that could fail if a `dataonly` volume is missing include:

- Reading or writing file data if the file's data extents were allocated from the missing `dataonly` volume.
- Using the `vxdump` command.

Volume availability is supported only on a file system with disk layout Version 7 or later.

Note: Do not mount a multi-volume system with the `ioerror=disable` or `ioerror=wdisable` mount options if the volumes have different availability properties. Symantec recommends the `ioerror=mdisable` mount option both for cluster mounts and for local mounts.

Creating multi-volume file systems

When a multi-volume file system is created, all volumes are `dataonly`, except volume zero, which is used to store the file system's metadata. The volume availability flag of volume zero cannot be set to `dataonly`.

As metadata cannot be allocated from `dataonly` volumes, enough metadata space should be allocated using `metadataok` volumes. The "file system out of space" error occurs if there is insufficient metadata space available, even if the `df` command shows that there is free space in the file system. The `fsvoladm` command can be used to see the free space in each volume and set the availability flag of the volume.

Unless otherwise specified, VxFS commands function the same on multi-volume file systems as the commands do on single-volume file systems.

Example of creating a multi-volume file system

The following procedure is an example of creating a multi-volume file system.

To create a multi-volume file system

- 1 After a volume set is created, create a VxFS file system by specifying the volume set name as an argument to `mkfs`:

```
# mkfs -V vxfs /dev/vx/rdisk/rootdg/myvset
version 7 layout
327680 sectors, 163840 blocks of size 1024,
log size 1024 blocks largefiles supported
```

After the file system is created, VxFS allocates space from the different volumes within the volume set.

- 2 List the component volumes of the volume set using of the `fsvoladm` command:

```
# mount -V vxfs /dev/vx/dsk/rootdg/myvset /mnt1
# fsvoladm list /mnt1
```

devid	size	used	avail	name
0	10240	1280	8960	vol1
1	51200	16	51184	vol2
2	51200	16	51184	vol3
3	51200	16	51184	vol4

- 3 Add a new volume by adding the volume to the volume set, then adding the volume to the file system:

```
# vxassist -g dg1 make vol5 50m
# vxvset -g dg1 addvol myvset vol5
# fsvoladm add /mnt1 vol5 50m
# fsvoladm list /mnt1
```

devid	size	used	avail	name
0	10240	1300	8940	vol1
1	51200	16	51184	vol2
2	51200	16	51184	vol3
3	51200	16	51184	vol4
4	51200	16	51184	vol5

- 4 List the volume availability flags using the `fsvoladm` command:

```
# fsvoladm queryflags /mnt1
```

volname	flags
vol1	metadataok
vol2	dataonly
vol3	dataonly
vol4	dataonly
vol5	dataonly

- 5 Increase the metadata space in the file system using the `fsvoladm` command:

```
# fsvoladm clearflags dataonly /mnt1 vol2
# fsvoladm queryflags /mnt1
```

volname	flags
vol1	metadataok
vol2	metadataok
vol3	dataonly
vol4	dataonly
vol5	dataonly

Converting a single volume file system to a multi-volume file system

The following procedure converts a traditional, single volume file system, `/mnt1`, on a single volume `vol1` in the diskgroup `dg1` to a multi-volume file system.

To convert a single volume file system

- 1 Determine the version of the volume's diskgroup:

```
# vxdg list dg1 | grep version: | awk '{ print $2 }'  
105
```

- 2 If the version is less than 110, upgrade the diskgroup:

```
# vxdg upgrade dg1
```

- 3 Determine the disk layout version of the file system:

```
# vxupgrade /mnt1  
Version 6
```

- 4 If the disk layout version is 6, upgrade to Version 7:

```
# vxupgrade -n 7 /mnt1
```

- 5 Unmount the file system:

```
# umount /mnt1
```

- 6 Convert the volume into a volume set:

```
# vxvset -g dg1 make vset1 vol1
```

- 7 Edit the `/etc/filesystems` file to replace the volume device name, `vol1`, with the volume set name, `vset1`.

- 8 Mount the file system:

```
# mount -V vxfs /dev/vx/dsk/dg1/vset1 /mnt1
```

- 9 As necessary, create and add volumes to the volume set:

```
# vxassist -g dg1 make vol2 256M  
# vxvset -g dg1 addvol vset1 vol2
```

- 10 Set the placement class tags on all volumes that do not have a tag:

```
# vxassist -g dg1 settag vol1 vxfs.placement_class.tier1  
# vxassist -g dg1 settag vol2 vxfs.placement_class.tier2
```

Adding a volume to and removing a volume from a multi-volume file system

You can add volumes to and remove volumes from a multi-volume file system using the `fsvoladm` command.

Adding a volume to a multi-volume file system

Use the `fsvoladm add` command to add a volume to a multi-volume file system.

To add a volume to a multi-volume file system

- ◆ Add a new volume to a multi-volume file system:

```
# fsvoladm add /mnt1 vol2 256m
```

Removing a volume from a multi-volume file system

Use the `fsvoladm remove` command to remove a volume from a multi-volume file system. The `fsvoladm remove` command fails if the volume being removed is the only volume in any allocation policy.

To remove a volume from a multi-volume file system

- ◆ Remove a volume from a multi-volume file system:

```
# fsvoladm remove /mnt1 vol2
```

Forcibly removing a volume

If you must forcibly remove a volume from a file system, such as if a volume is permanently destroyed and you want to clean up the dangling pointers to the lost volume, use the `fsck -o zapvol=volname` command. The `zapvol` option performs a full file system check and zaps all inodes that refer to the specified volume. The `fsck` command prints the inode numbers of all files that the command destroys; the file names are not printed. The `zapvol` option only affects regular files if used on a `dataonly` volume. However, it could destroy structural files if used on a `metadataok` volume, which can make the file system unrecoverable. Therefore, the `zapvol` option should be used with caution on `metadataok` volumes.

Moving volume 0

Volume 0 in a multi-volume file system cannot be removed from the file system, but you can move volume 0 to different storage using the `vxassist move` command. The `vxassist` command creates any necessary temporary mirrors and cleans up the mirrors at the end of the operation.

To move volume 0

- ◆ Move volume 0:

```
# vxassist -g mydg move vol1 \!mydg
```

Volume encapsulation

Multi-volume support enables the ability to encapsulate an existing raw volume and make the volume contents appear as a file in the file system.

Encapsulating a volume involves the following actions:

- Adding the volume to an existing volume set.
- Adding the volume to the file system using `fsvoladm`.

Encapsulating a volume

The following example illustrates how to encapsulate a volume.

To encapsulate a volume

- 1 List the volumes:

```
# vxvset -g dg1 list myvset
VOLUME  INDEX  LENGTH  STATE  CONTEXT
vol1     0       102400  ACTIVE -
vol2     1       102400  ACTIVE -
```

The volume set has two volumes.

- 2 Create a third volume and copy the passwd file to the third volume:

```
# vxassist -g dg1 make dbvol 100m
# dd if=/etc/passwd of=/dev/vx/rdisk/rootdg/dbvol count=1
1+0 records in
1+0 records out
```

The third volume will be used to demonstrate how the volume can be accessed as a file, as shown later.

3 Create a file system on the volume set:

```
# mkfs -V vxfs /dev/vx/rdisk/rootdg/myvset
version 7 layout
204800 sectors, 102400 blocks of size 1024,
log size 1024 blocks
largefiles supported
```

4 Mount the volume set:

```
# mount -V vxfs /dev/vx/dsk/rootdg/myvset /mnt1
```

5 Add the new volume to the volume set:

```
# vxvset -g dg1 addvol myvset dbvol
```

6 Encapsulate dbvol:

```
# fsvoladm encapsulate /mnt1/dbfile dbvol 100m
# ls -l /mnt1/dbfile
-rw----- 1 root other 104857600 May 22 11:30 /mnt1/dbfile
```

7 Examine the contents of dbfile to see that it can be accessed as a file:

```
# head -2 /mnt1/dbfile
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1:/:
```

The passwd file that was written to the raw volume is now visible in the new file.

Note: If the encapsulated file is changed in any way, such as if the file is extended, truncated, or moved with an allocation policy or resized volume, or the volume is encapsulated with a bias, the file cannot be de-encapsulated.

Deencapsulating a volume

The following example illustrates how to deencapsulate a volume.

To deencapsulate a volume**1 List the volumes:**

```
# vxvset -g dgl list myvset
VOLUME   INDEX   LENGTH   STATE   CONTEXT
vol1     0       102400   ACTIVE  -
vol2     1       102400   ACTIVE  -
dbvol    2       102400   ACTIVE  -
```

The volume set has three volumes.

2 Deencapsulate dbvol:

```
# fsvoladm deencapsulate /mnt1/dbfile
```

Reporting file extents

MVS feature provides the capability for file-to-volume mapping and volume-to-file mapping via the `fsmmap` and `fsvmap` commands. The `fsmmap` command reports the volume name, logical offset, and size of data extents, or the volume name and size of indirect extents associated with a file on a multi-volume file system. The `fsvmap` command maps volumes to the files that have extents on those volumes.

See the `fsmmap(1M)` and `fsvmap(1M)` manual pages.

The `fsmmap` command requires `open()` permission for each file or directory specified. Root permission is required to report the list of files with extents on a particular volume.

Examples of reporting file extents

The following examples show typical uses of the `fsmmap` and `fsvmap` commands.

Using the fsmmap command

- ◆ Use the `find` command to descend directories recursively and run `fsmmap` on the list of files:

```
# find . | fsmmap -
Volume Extent Type   File
vol2   Data                ./file1
vol1   Data                ./file2
```

Using the `fsvmap` command

- 1 Report the extents of files on multiple volumes:

```
# fsvmap /dev/vx/rdisk/fstest/testvset vol1 vol2
vol1 /.
vol1 /ns2
vol1 /ns3
vol1 /file1
vol2 /file1
vol2 /file2
```

- 2 Report the extents of files that have either data or metadata on a single volume in all Storage Checkpoints, and indicate if the volume has file system metadata:

```
# fsvmap -mVC /dev/vx/rdisk/fstest/testvset vol1
Meta Structural vol1 //volume has filesystem metadata//
Data UNNAMED vol1 /.
Data UNNAMED vol1 /ns2
Data UNNAMED vol1 /ns3
Data UNNAMED vol1 /file1
Meta UNNAMED vol1 /file1
```

Load balancing

An allocation policy with the balance allocation order can be defined and assigned to files that must have their allocations distributed at random between a set of specified volumes. Each extent associated with these files are limited to a maximum size that is defined as the required chunk size in the allocation policy. The distribution of the extents is mostly equal if none of the volumes are full or disabled.

Load balancing allocation policies can be assigned to individual files or for all files in the file system. Although intended for balancing data extents across volumes, a load balancing policy can be assigned as a metadata policy if desired, without any restrictions.

Note: If a file has both a fixed extent size set and an allocation policy for load balancing, certain behavior can be expected. If the chunk size in the allocation policy is greater than the fixed extent size, all extents for the file are limited by the chunk size. For example, if the chunk size is 16 MB and the fixed extent size is 3 MB, then the largest extent that satisfies both the conditions is 15 MB. If the fixed extent size is larger than the chunk size, all extents are limited to the fixed extent size. For example, if the chunk size is 2 MB and the fixed extent size is 3 MB, then all extents for the file are limited to 3 MB.

Defining and assigning a load balancing allocation policy

The following example defines a load balancing policy and assigns the policy to the file, `/mnt/file.db`.

To define and assign the policy

- 1 Define the policy by specifying the `-o balance` and `-c` options:

```
# fsapadm define -o balance -c 2m /mnt loadbal vol1 vol2 vol3 vol4
```

- 2 Assign the policy:

```
# fsapadm assign /mnt/filedb loadbal meta
```

Rebalancing extents

Extents can be rebalanced by strictly enforcing the allocation policy. Rebalancing is generally required when volumes are added or removed from the policy or when the chunk size is modified. When volumes are removed from the volume set, any extents on the volumes being removed are automatically relocated to other volumes within the policy.

The following example redefines a policy that has four volumes by adding two new volumes, removing an existing volume, and enforcing the policy for rebalancing.

To rebalance extents

- 1 Define the policy by specifying the `-o balance` and `-c` options:

```
# fsapadm define -o balance -c 2m /mnt loadbal vol1 vol2 vol4 \  
vol5 vol6
```

- 2 Enforce the policy:

```
# fsapadm enforcefile -f strict /mnt/filedb
```

Converting a multi-volume file system to a single volume file system

Because data can be relocated among volumes in a multi-volume file system, you can convert a multi-volume file system to a traditional, single volume file system by moving all file system data onto a single volume. Such a conversion is useful to users who would like to try using a multi-volume file system or SmartTier, but are not committed to using a multi-volume file system permanently.

See [“About SmartTier”](#) on page 337.

There are three restrictions to this operation:

- The single volume must be the first volume in the volume set
- The first volume must have sufficient space to hold all of the data and file system metadata
- The volume cannot have any allocation policies that restrict the movement of data

Converting to a single volume file system

The following procedure converts an existing multi-volume file system, `/mnt1`, of the volume set `vset1`, to a single volume file system, `/mnt1`, on volume `vol1` in diskgroup `dg1`.

Note: Steps 5, 6, 7, and 8 are optional, and can be performed if you prefer to remove the wrapper of the volume set object.

Converting to a single volume file system

- 1 Determine if the first volume in the volume set, which is identified as device number 0, has the capacity to receive the data from the other volumes that will be removed:

```
# df /mnt1
/mnt1 (/dev/vx/dsk/dg1/vol1):16777216 blocks 3443528 files
```

- 2 If the first volume does not have sufficient capacity, grow the volume to a sufficient size:

```
# fsvoladm resize /mnt1 vol1 150g
```

- 3 Remove all existing allocation policies:

```
# fsppadm unassign /mnt1
```

- 4 Remove all volumes except the first volume in the volume set:

```
# fsvoladm remove /mnt1 vol2
# vxvset -g dg1 rmvol vset1 vol2
# fsvoladm remove /mnt1 vol3
# vxvset -g dg1 rmvol vset1 vol3
```

Before removing a volume, the file system attempts to relocate the files on that volume. Successful relocation requires space on another volume, and no allocation policies can be enforced that pin files to that volume. The time for the command to complete is proportional to the amount of data that must be relocated.

- 5 Unmount the file system:

```
# umount /mnt1
```

- 6 Remove the volume from the volume set:

```
# vxvset -g dg1 rmvol vset1 vol1
```

- 7 Edit the `/etc/filesystems` file to replace the volume set name, `vset1`, with the volume device name, `vol1`.

- 8 Mount the file system:

```
# mount -V vxfs /dev/vx/dsk/dg1/vol1 /mnt1
```

Administering SmartTier

This chapter includes the following topics:

- [About SmartTier](#)
- [Supported SmartTier document type definitions](#)
- [Placement classes](#)
- [Administering placement policies](#)
- [File placement policy grammar](#)
- [File placement policy rules](#)
- [Calculating I/O temperature and access temperature](#)
- [Multiple criteria in file placement policy rule statements](#)
- [File placement policy rule and statement ordering](#)
- [File placement policies and extending files](#)
- [Using SmartTier with solid state disks](#)

About SmartTier

VxFS uses multi-tier online storage by way of the SmartTier feature, which functions on top of multi-volume file systems. Multi-volume file systems are file systems that occupy two or more virtual volumes. The collection of volumes is known as a volume set. A volume set is made up of disks or disk array LUNs belonging to a single Veritas Volume Manager (VxVM) disk group. A multi-volume file system presents a single name space, making the existence of multiple volumes transparent to users and applications. Each volume retains a separate identity

for administrative purposes, making it possible to control the locations to which individual files are directed.

See “[About multi-volume support](#)” on page 323.

Note: Some of the commands have changed or been removed between the 4.1 release and the 5.1 SP1 release to make placement policy management more user-friendly. The following commands have been removed: `fsrpadm`, `fsmove`, and `fssweep`. The output of the `queryfile`, `queryfs`, and `list` options of the `fsapadm` command now print the allocation order by name instead of number.

In the previous VxFS 5.x releases, SmartTier was known as Dynamic Storage Tiering.

SmartTier allows administrators of multi-volume VxFS file systems to manage the placement of files and the placement of portions of files on individual volumes in a volume set by defining placement policies. Placement policies control both initial file location and the circumstances under which existing files are relocated. These placement policies cause the files to which they apply to be created and extended on specific subsets of a file system's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet the specified naming, timing, access rate, and storage capacity-related conditions.

You make a VxVM volume part of a placement class by associating a volume tag with it. For file placement purposes, VxFS treats all of the volumes in a placement class as equivalent, and balances space allocation across them. A volume may have more than one tag associated with it. If a volume has multiple tags, the volume belongs to multiple placement classes and is subject to allocation and relocation policies that relate to any of the placement classes. Multiple tagging should be used carefully.

See “[Placement classes](#)” on page 339.

VxFS imposes no capacity, performance, availability, or other constraints on placement classes. Any volume may be added to any placement class, no matter what type the volume has nor what types other volumes in the class have. However, a good practice is to place volumes of similar I/O performance and availability in the same placement class.

The *Using SmartTier* Symantec Yellow Book provides additional information regarding the SmartTier feature, including the value of SmartTier and best practices for using SmartTier. You can download *Using SmartTier* from the following Web page:

<http://www.symantec.com/enterprise/yellowbooks/index.jsp>

Supported SmartTier document type definitions

Table 25-1 describes which releases of Veritas File System (VxFS) support specific SmartTier document type definitions (DTDs).

Table 25-1 Supported SmartTier document type definitions

VxFS Version	DTD Version	
	1.0	1.1
5.0	Supported	Not supported
5.1	Supported	Supported
5.1 SP1	Supported	Supported

Placement classes

A placement class is a SmartTier attribute of a given volume in a volume set of a multi-volume file system. This attribute is a character string, and is known as a volume tag. A volume can have different tags, one of which can be the placement class. The placement class tag makes a volume distinguishable by SmartTier.

Volume tags are organized as hierarchical name spaces in which periods separate the levels of the hierarchy. By convention, the uppermost level in the volume tag hierarchy denotes the Storage Foundation component or application that uses a tag, and the second level denotes the tag's purpose. SmartTier recognizes volume tags of the form `vxfs.placement_class.class_name`. The prefix `vxfs` identifies a tag as being associated with VxFS. The `placement_class` string identifies the tag as a file placement class that SmartTier uses. The `class_name` string represents the name of the file placement class to which the tagged volume belongs. For example, a volume with the tag `vxfs.placement_class.tier1` belongs to placement class `tier1`. Administrators use the `vxassist` command to associate tags with volumes.

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

VxFS policy rules specify file placement in terms of placement classes rather than in terms of individual volumes. All volumes that belong to a particular placement class are interchangeable with respect to file creation and relocation operations. Specifying file placement in terms of placement classes rather than in terms of specific volumes simplifies the administration of multi-tier storage.

The administration of multi-tier storage is simplified in the following ways:

- Adding or removing volumes does not require a file placement policy change. If a volume with a tag value of `vxfs.placement_class.tier2` is added to a file system's volume set, all policies that refer to `tier2` immediately apply to the newly added volume with no administrative action. Similarly, volumes can be evacuated, that is, have data removed from them, and be removed from a file system without a policy change. The active policy continues to apply to the file system's remaining volumes.
- File placement policies are not specific to individual file systems. A file placement policy can be assigned to any file system whose volume set includes volumes tagged with the tag values (placement classes) named in the policy. This property makes it possible for data centers with large numbers of servers to define standard placement policies and apply them uniformly to all servers with a single administrative action.

Tagging volumes as placement classes

The following example tags the `vsavola` volume as placement class `tier1`, `vsavolb` as placement class `tier2`, `vsavolc` as placement class `tier3`, and `vsavold` as placement class `tier4` using the `vxassist settag` command.

To tag volumes

- ◆ Tag the volumes as placement classes:

```
# vxassist -g cfsdg settag vsavola vxfs.placement_class.tier1
# vxassist -g cfsdg settag vsavolb vxfs.placement_class.tier2
# vxassist -g cfsdg settag vsavolc vxfs.placement_class.tier3
# vxassist -g cfsdg settag vsavold vxfs.placement_class.tier4
```

Listing placement classes

Placement classes are listed using the `vxassist listtag` command.

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

The following example lists all volume tags, including placement classes, set on a volume `vsavola` in the diskgroup `cfsdg`.

To list placement classes

- ◆ List the volume tags, including placement classes:

```
# vxassist -g cfsdg listtag vsavola
```

Administering placement policies

A VxFS file placement policy document contains rules by which VxFS creates, relocates, and deletes files, but the placement policy does not refer to specific file systems or volumes. You can create a file system's active file placement policy by assigning a placement policy document to the file system via the `fsppadm` command or the GUI.

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

At most, one file placement policy can be assigned to a VxFS file system at any time. A file system may have no file placement policy assigned to it, in which case VxFS allocates space for new files according to its own internal algorithms.

In systems with Storage Foundation Management Server (SFMS) software installed, file placement policy information is stored in the SFMS database. The SFMS database contains both XML policy documents and lists of hosts and file systems for which each document is the current active policy. When a policy document is updated, SFMS can assign the updated document to all file systems whose current active policies are based on that document. By default, SFMS does not update file system active policies that have been created or modified locally, that is by the hosts that control the placement policies' file systems. If a SFMS administrator forces assignment of a placement policy to a file system, the file system's active placement policy is overwritten and any local changes that had been made to the placement policy are lost.

You can view sample placement policies in the `/opt/VRTSvxfs/etc` directory. These sample placement policies are installed as part of the VxFS package installation.

Assigning a placement policy

The following example uses the `fsppadm assign` command to assign the file placement policy represented in the XML policy document `/tmp/policy1.xml` for the file system at mount point `/mnt1`.

To assign a placement policy

- ◆ Assign a placement policy to a file system:

```
# fsppadm assign /mnt1 /tmp/policy1.xml
```

Unassigning a placement policy

The following example uses the `fsppadm unassign` command to unassign the active file placement policy from the file system at mount point `/mnt1`.

To unassign a placement policy

- ◆ Unassign the placement policy from a file system:

```
# fspadm unassign /mnt1
```

Analyzing the space impact of enforcing a placement policy

The following example uses the `fspadm analyze` command to analyze the impact if the enforce operation is performed on the file placement policy represented in the XML policy document `/tmp/policy1.xml` for the mount point `/mnt1`. The command builds the I/O temperature database if necessary.

To analyze the space impact of enforcing a placement policy

- ◆ Analyze the impact of enforcing the file placement policy represented in the XML policy document `/tmp/policy1.xml` for the mount point `/mnt1`:

```
# fspadm analyze -F /tmp/policy1.xml -i /mnt1
```

Querying which files will be affected by enforcing a placement policy

The following example uses the `fspadm query` command to generate a list of files that will be affected by enforcing a placement policy. The command provides details about where the files currently reside, to where the files will be relocated, and which rule in the placement policy applies to the files.

To query which files will be affected by enforcing a placement policy

- ◆ Query the files that will be affected:

```
# fspadm query /mnt1/dir1/dir2 /mnt2 /mnt1/dir3
```

Enforcing a placement policy

Enforcing a placement policy for a file system requires that the policy be assigned to the file system. You must assign a placement policy before it can be enforced.

See [“Assigning a placement policy”](#) on page 341.

Enforce operations are logged in a hidden file, `._fspadm_enforce.log`, in the `lost+found` directory of the mount point. This log file contains details such as files' previous locations, the files' new locations, and the reasons for the files' relocations. The enforce operation creates the `._fspadm_enforce.log` file if the file does not exist. The enforce operation appends the file if the file already

exists. The `._fsspadm_enforce.log` file can be backed up or removed as with a normal file.

You can specify the `-F` option to specify a placement policy other than the existing active placement policy. This option can be used to enforce the rules given in the specified placement policy for maintenance purposes, such as for reclaiming a LUN from the file system.

You can specify the `-p` option to specify the number of concurrent threads to be used to perform the `fsspadm` operation. You specify the `io_nice` parameter as an integer between 1 and 100, with 50 being the default value. A value of 1 specifies 1 slave and 1 master thread per mount. A value of 50 specifies 16 slaves and 1 master thread per mount. A value of 100 specifies 32 slaves and 1 master thread per mount.

You can specify the `-C` option so that the `fsspadm` command processes only those files that have some activity stats logged in the File Change Log (FCL) file during the period specified in the placement policy. You can use the `-C` option only if the policy's `ACCESSTEMP` or `IOTEMP` elements use the `Prefer` criteria.

You can specify the `-T` option to specify the placement classes that contain files for the `fsspadm` command to sweep and relocate selectively. You can specify the `-T` option only if the policy uses the `Prefer` criteria for `IOTEMP`.

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

The following example uses the `fsspadm enforce` command to enforce the file placement policy for the file system at mount point `/mnt1`, and includes the access time, modification time, and file size of the specified paths in the report, `/tmp/report`.

To enforce a placement policy

- ◆ Enforce a placement policy for a file system:

```
# fspadm enforce -a -r /tmp/report /mnt1
Current Current Relocated Relocated
Class Volume Class Volume Rule File
tier3 vole tier3 vole a_to_z /mnt1/mds1/d1/file1
tier3 vole tier3 vole a_to_z /mnt1/mds1/d1/file2
tier3 vole tier3 vole a_to_z /mnt1/mds1/d1/d2/file3
tier3 volf tier3 volf a_to_z /mnt1/mds1/d1/d2/file4
.
.
.
Sweep path : /mnt1
Files moved : 42
KB moved : 1267
```

Tier Name	Size (KB)	Free Before (KB)	Free After (KB)
tier4	524288	524256	524256
tier3	524288	522968	522968
tier2	524288	524256	524256
tier1	524288	502188	501227

Validating a placement policy

The following example uses the `fspadm validate` command to validate the placement policy `policy.xml` against all mounted file systems.

To validate a placement policy against all mounted file systems

- ◆ Validate the placement policy:

```
# fspadm validate /tmp/policy.xml
```

File placement policy grammar

VxFS allocates and relocates files within a multi-volume file system based on properties in the file system metadata that pertains to the files. Placement decisions may be based on file name, directory of residence, time of last access, access frequency, file size, and ownership. An individual file system's criteria for

allocating and relocating files are expressed in the file system's file placement policy.

A VxFS file placement policy defines the desired placement of sets of files on the volumes of a VxFS multi-volume file system. A file placement policy specifies the placement classes of volumes on which files should be created, and where and under what conditions the files should be relocated to volumes in alternate placement classes or deleted. You can create file placement policy documents, which are XML text files, using an XML editor, a text editor, or Veritas Operations Manager (VOM).

See the `/opt/VRTSvxfs/etc/placement_policy.dtd` file for the overall structure of a placement policy.

File placement policy rules

A VxFS file placement policy consists of one or more rules. Each rule applies to one or more files. The files to which a rule applies are designated in one or more `SELECT` statements. A `SELECT` statement designates files according to one or more of four properties: their names or naming patterns, the directories in which they reside, their owners' user names, and their owners' group names.

A file may be designated by more than one rule. For example, if one rule designates files in directory `/dir`, and another designates files owned by `user1`, a file in `/dir` that is owned by `user1` is designated by both rules. Only the rule that appears first in the placement policy applies to the file; subsequent rules are ignored.

You can define placement policies that do not encompass the entire file system name space. When a file that is not designated by any rule in its file system's active placement policy is created, VxFS places the file according to its own internal algorithms. To maintain full control over file placement, include a catchall rule at the end of each placement policy document with a `SELECT` statement that designates files by the naming pattern `*`. Such a rule designates all files that have not been designated by the rules appearing earlier in the placement policy document.

Two types of rules exist: `data` and `ckpt`. The `data` rule type allows SmartTier to relocate normal data files. The `ckpt` rule type allows SmartTier to relocate Storage Checkpoints. You specify the rule type by setting the `Flags` attribute for the rule.

SELECT statement

The VxFS placement policy rule `SELECT` statement designates the collection of files to which a rule applies.

The following XML snippet illustrates the general form of the `SELECT` statement:

```
<SELECT>
  <DIRECTORY Flags="directory_flag_value"> value
</DIRECTORY>
  <PATTERN Flags="pattern_flag_value"> value </PATTERN>
  <USER> value </USER>
  <GROUP> value </GROUP>
</SELECT>
```

A `SELECT` statement may designate files by using the following selection criteria:

`<DIRECTORY>` A full path name relative to the file system mount point. The `Flags="directory_flag_value"` XML attribute must have a value of `nonrecursive`, denoting that only files in the specified directory are designated, or a value of `recursive`, denoting that files in all subdirectories of the specified directory are designated. The `Flags` attribute is mandatory.

The `<DIRECTORY>` criterion is optional, and may be specified more than once.

<PATTERN> Either an exact file name or a pattern using a single wildcard character (*). The first "*" character is treated as a wildcard, while any subsequent "*" characters are treated as literal text. The pattern cannot contain "/".

The following list contains examples of patterns:

- abc* – Matches all files whose names begin with “abc”.
- abc.* – Matches all files whose names are exactly “abc” followed by a period and any extension.
- *abc – Matches all files whose names end in “abc”, even if the name is all or part of an extension.
- *.abc – Matches files of any name whose name extension (following the period) is “abc”.
- ab*c – Matches all files whose names start with “ab” and end with “c”.

The wildcard character matches any character, including ".", "?", and "[", unlike using the wildcard in a shell.

The `Flags="pattern_flag_value"` XML attribute is optional, and if specified can only have a value of `recursive`. Specify `Flags="recursive"` only if the pattern is a directory. If `Flags` is not specified, the default attribute value is `nonrecursive`. If `Flags="recursive"` is specified, the enclosing selection criteria selects all files in any component directory that is anywhere below the directory specified by `<DIRECTORY>` if the component directory matches the pattern and either of the following is true:

- `<DIRECTORY>` is specified and has the recursive flag.
- `<DIRECTORY>` is not specified and the directory is anywhere in the file system.

If the pattern contains the wildcard character (*), wildcard character matching is performed.

The `<PATTERN>` criterion is optional, and may be specified more than once. Only one value can be specified per `<PATTERN>` element.

<USER> User name of the file's owner. The user number cannot be specified in place of the name.

The `<USER>` criterion is optional, and may be specified more than once.

<GROUP> Group name of the file's owner. The group number cannot be specified in place of the group name.

The `<GROUP>` criterion is optional, and may be specified more than once.

One or more instances of any or all of the file selection criteria may be specified within a single `SELECT` statement. If two or more selection criteria of different types are specified in a single statement, a file must satisfy one criterion of each type to be selected.

In the following example, only files that reside in either the `ora/db` or the `crash/dump` directory, and whose owner is either `user1` or `user2` are selected for possible action:

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">ora/db</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">crash/dump</DIRECTORY>
  <USER>user1</USER>
  <USER>user2</USER>
</SELECT>
```

A rule may include multiple `SELECT` statements. If a file satisfies the selection criteria of one of the `SELECT` statements, it is eligible for action.

In the following example, any files owned by either `user1` or `user2`, no matter in which directories they reside, as well as all files in the `ora/db` or `crash/dump` directories, no matter which users own them, are eligible for action:

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">ora/db</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">crash/dump</DIRECTORY>
</SELECT>
<SELECT>
  <USER>user1</USER>
  <USER>user2</USER>
</SELECT>
```

When VxFS creates new files, VxFS applies active placement policy rules in the order of appearance in the active placement policy's XML source file. The first rule in which a `SELECT` statement designates the file to be created determines the file's placement; no later rules apply. Similarly, VxFS scans the active policy rules on behalf of each file when relocating files, stopping the rules scan when it reaches the first rule containing a `SELECT` statement that designates the file. This behavior holds true even if the applicable rule results in no action. Take for example a policy rule that indicates that `.dat` files inactive for 30 days should be relocated, and a later rule indicates that `.dat` files larger than 10 megabytes should be relocated. A 20 megabyte `.dat` file that has been inactive for 10 days will not be relocated because the earlier rule applied. The later rule is never scanned.

A placement policy rule's action statements apply to all files designated by any of the rule's `SELECT` statements. If an existing file is not designated by a `SELECT` statement in any rule of a file system's active placement policy, then SmartTier does not relocate or delete the file. If an application creates a file that is not designated by a `SELECT` statement in a rule of the file system's active policy, then VxFS places the file according to its own internal algorithms. If this behavior is inappropriate, the last rule in the policy document on which the file system's active placement policy is based should specify `<PATTERN>*</PATTERN>` as the only selection criterion in its `SELECT` statement, and a `CREATE` statement naming the desired placement class for files not selected by other rules.

CREATE statement

A `CREATE` statement in a file placement policy rule specifies one or more placement classes of volumes on which VxFS should allocate space for new files to which the rule applies at the time the files are created. You can specify only placement classes, not individual volume names, in a `CREATE` statement.

A file placement policy rule may contain at most one `CREATE` statement. If a rule does not contain a `CREATE` statement, VxFS places files designated by the rule's `SELECT` statements according to its internal algorithms. However, rules without `CREATE` statements can be used to relocate or delete existing files that the rules' `SELECT` statements designate.

The following XML snippet illustrates the general form of the `CREATE` statement:

```
<CREATE>
  <ON Flags="flag_value">
    <DESTINATION>
      <CLASS> placement_class_name </CLASS>
      <BALANCE_SIZE Units="units_specifier"> chunk_size
    </BALANCE_SIZE>
    </DESTINATION>
    <DESTINATION> additional_placement_class_specifications
  </DESTINATION>
  </ON>
</CREATE>
```

A `CREATE` statement includes a single `<ON>` clause, in which one or more `<DESTINATION>` XML elements specify placement classes for initial file allocation in order of decreasing preference. VxFS allocates space for new files to which a rule applies on a volume in the first class specified, if available space permits. If space cannot be allocated on any volume in the first class, VxFS allocates space on a volume in the second class specified if available space permits, and so forth.

If space cannot be allocated on any volume in any of the placement classes specified, file creation fails with an `ENOSPC` error, even if adequate space is available elsewhere in the file system's volume set. This situation can be circumvented by specifying a `Flags` attribute with a value of "any" in the `<ON>` clause. If `<ON Flags="any">` is specified in a `CREATE` statement, VxFS first attempts to allocate space for new files to which the rule applies on the specified placement classes. Failing that, VxFS resorts to its internal space allocation algorithms, so file allocation does not fail unless there is no available space any-where in the file system's volume set.

The `Flags="any"` attribute differs from the catchall rule in that this attribute applies only to files designated by the `SELECT` statement in the rule, which may be less inclusive than the `<PATTERN>*/</PATTERN>` file selection specification of the catchall rule.

In addition to the placement class name specified in the `<CLASS>` sub-element, a `<DESTINATION>` XML element may contain a `<BALANCE_SIZE>` sub-element. Presence of a `<BALANCE_SIZE>` element indicates that space allocation should be distributed across the volumes of the placement class in chunks of the indicated size. For example, if a balance size of one megabyte is specified for a placement class containing three volumes, VxFS allocates the first megabyte of space for a new or extending file on the first (lowest indexed) volume in the class, the second megabyte on the second volume, the third megabyte on the third volume, the fourth megabyte on the first volume, and so forth. Using the `Units` attribute in the `<BALANCE_SIZE>` XML tag, the balance size value may be specified in the following units:

bytes	Bytes
KB	Kilobytes
MB	Megabytes
GB	Gigabytes

The `<BALANCE_SIZE>` element distributes the allocation of database files across the volumes in a placement class. In principle, distributing the data in each file across multiple volumes distributes the I/O load across the volumes as well.

The `CREATE` statement in the following example specifies that files to which the rule applies should be created on the `tier1` volume if space is available, and on one of the `tier2` volumes if not. If space allocation on `tier1` and `tier2` volumes is not possible, file creation fails, even if space is available on `tier3` volumes.

```

<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier2</CLASS>
      <BALANCE_SIZE Units="MB">1</BALANCE_SIZE>
    </DESTINATION>
  </ON>
</CREATE>

```

The `<BALANCE_SIZE>` element with a value of one megabyte is specified for allocations on `tier2` volumes. For files allocated on `tier2` volumes, the first megabyte would be allocated on the first volume, the second on the second volume, and so forth.

RELOCATE statement

The `RELOCATE` action statement of file placement policy rules specifies an action that VxFS takes on designated files during periodic scans of the file system, and the circumstances under which the actions should be taken. The `fspadm enforce` command is used to scan all or part of a file system for files that should be relocated based on rules in the active placement policy at the time of the scan.

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

The `fspadm enforce` command scans file systems in path name order. For each file, VxFS identifies the first applicable rule in the active placement policy, as determined by the rules' `SELECT` statements. If the file resides on a volume specified in the `<FROM>` clause of one of the rule's `RELOCATE` statements, and if the file meets the criteria for relocation specified in the statement's `<WHEN>` clause, the file is scheduled for relocation to a volume in the first placement class listed in the `<TO>` clause that has space available for the file. The scan that results from issuing the `fspadm enforce` command runs to completion before any files are relocated.

The following XML snippet illustrates the general form of the `RELOCATE` statement:

```

<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE> additional_placement_class_specifications
    </SOURCE>

```

```

</FROM>
<TO>
  <DESTINATION>
    <CLASS> placement_class_name </CLASS>
    <BALANCE_SIZE Units="units_specifier">
      chunk_size
    </BALANCE_SIZE>
  </DESTINATION>
  <DESTINATION>
    additional_placement_class_specifications
  </DESTINATION>
</TO>
<WHEN> relocation_conditions </WHEN>
</RELOCATE>

```

A RELOCATE statement contains the following clauses:

<FROM> An optional clause that contains a list of placement classes from whose volumes designated files should be relocated if the files meet the conditions specified in the <WHEN> clause. No priority is associated with the ordering of placement classes listed in a <FROM> clause. If a file to which the rule applies is located on a volume in any specified placement class, the file is considered for relocation.

If a RELOCATE statement contains a <FROM> clause, VxFS only considers files that reside on volumes in placement classes specified in the clause for relocation. If no <FROM> clause is present, qualifying files are relocated regardless of where the files reside.

<TO> Indicates the placement classes to which qualifying files should be relocated. Unlike the source placement class list in a FROM clause, placement classes in a <TO> clause are specified in priority order. Files are relocated to volumes in the first specified placement class if possible, to the second if not, and so forth.

The <TO> clause of the RELOCATE statement contains a list of <DESTINATION> XML elements specifying placement classes to whose volumes VxFS relocates qualifying files. Placement classes are specified in priority order. VxFS relocates qualifying files to volumes in the first placement class specified as long as space is available. A <DESTINATION> element may contain an optional <BALANCE_SIZE> modifier sub-element. The <BALANCE_SIZE> modifier indicates that relocated files should be distributed across the volumes of the destination placement class in chunks of the indicated size. For example, if a balance size of one megabyte is specified for a placement class containing three volumes, VxFS relocates the first megabyte the file to the first (lowest indexed) volume in the class, the second megabyte to the second volume, the third megabyte to the third volume, the fourth megabyte to the first volume, and so forth. Using the Units attribute in the <BALANCE_SIZE> XML tag, the chunk value may be specified in the balance size value may be specified in bytes (Units="bytes"), kilobytes (Units="KB"), megabytes (Units="MB"), or gigabytes (Units="GB").

The <BALANCE_SIZE> element distributes the allocation of database files across the volumes in a placement class. In principle, distributing the data in each file across multiple volumes distributes the I/O load across the volumes as well.

<WHEN> An optional clause that indicates the conditions under which files to which the rule applies should be relocated. Files that have been unaccessed or unmodified for a specified period, reached a certain size, or reached a specific I/O temperature or access temperature level may be relocated. If a RELOCATE statement does not contain a <WHEN> clause, files to which the rule applies are relocated unconditionally.

A <WHEN> clause may be included in a RELOCATE statement to specify that files should be relocated only if any or all of four types of criteria are met. Files can be specified for relocation if they satisfy one or more criteria.

The following are the criteria that can be specified for the <WHEN> clause:

<ACCAGE> This criterion is met when files are inactive for a designated period or during a designated period relative to the time at which the `fsppadm enforce` command was issued.

<MODAGE>	This criterion is met when files are unmodified for a designated period or during a designated period relative to the time at which the <code>fsppadm enforce</code> command was issued.
<SIZE>	This criterion is met when files exceed or drop below a designated size or fall within a designated size range.
<IOTEMP>	This criterion is met when files exceed or drop below a designated I/O temperature, or fall within a designated I/O temperature range. A file's I/O temperature is a measure of the I/O activity against it during the period designated by the <PERIOD> element prior to the time at which the <code>fsppadm enforce</code> command was issued. See “Calculating I/O temperature and access temperature” on page 367.
<ACCESSTEMP>	This criterion is met when files exceed or drop below a specified average access temperature, or fall within a specified access temperature range. A file's access temperature is similar to its I/O temperature, except that access temperature is computed using the number of I/O requests to the file, rather than the number of bytes transferred.

Note: The use of <IOTEMP> and <ACCESSTEMP> for data placement on VxFS servers that are used as NFS servers may not be very effective due to NFS caching. NFS client side caching and the way that NFS works can result in I/O initiated from an NFS client not producing NFS server side I/O. As such, any temperature measurements in place on the server side will not correctly reflect the I/O behavior that is specified by the placement policy.

If the server is solely used as an NFS server, this problem can potentially be mitigated by suitably adjusting or lowering the temperature thresholds. However, adjusting the thresholds may not always create the desired effect. In addition, if the same mount point is used both as an NFS export as well as a local mount, the temperature-based placement decisions will not be very effective due to the NFS cache skew.

The following XML snippet illustrates the general form of the <WHEN> clause in a RELOCATE statement:

```
<WHEN>
  <ACCAGE Units="units_value">
    <MIN Flags="comparison_operator">
      min_access_age</MIN>
    <MAX Flags="comparison_operator">
```

```

        max_access_age</MAX>
</ACCAGE>
<MODAGE Units="units_value">
    <MIN Flags="comparison_operator">
        min_modification_age</MIN>
    <MAX Flags="comparison_operator">
        max_modification_age</MAX>
</MODAGE>
<SIZE " Units="units_value">
    <MIN Flags="comparison_operator">
        min_size</MIN>
    <MAX Flags="comparison_operator">
        max_size</MAX>
</SIZE>
<IOTEMP Type="read_write_preference" Prefer="temperature_preference">
    <MIN Flags="comparison_operator">
        min_I/O_temperature</MIN>
    <MAX Flags="comparison_operator">
        max_I/O_temperature</MAX>
    <PERIOD Units="days_or_hours"> days_or_hours_of_interest </PERIOD>
</IOTEMP>
<ACCESSTEMP Type="read_write_preference"
Prefer="temperature_preference">
    <MIN Flags="comparison_operator">
        min_access_temperature</MIN>
    <MAX Flags="comparison_operator">
        max_access_temperature</MAX>
    <PERIOD Units="days_or_hours"> days_or_hours_of_interest </PERIOD>
</ACCESSTEMP>
</WHEN>

```

The access age (<ACCAGE>) element refers to the amount of time since a file was last accessed. VxFS computes access age by subtracting a file's time of last access, atime, from the time when the `fsppadm enforce` command was issued. The <MIN> and <MAX> XML elements in an <ACCAGE> clause, denote the minimum and maximum access age thresholds for relocation, respectively. These elements are optional, but at least one must be included. Using the `Units` XML attribute, the <MIN> and <MAX> elements may be specified in the following units:

hours	Hours
days	Days. A day is considered to be 24 hours prior to the time that the <code>fsppadm enforce</code> command was issued.

Both the `<MIN>` and `<MAX>` elements require `Flags` attributes to direct their operation.

For `<MIN>`, the following `Flags` attributes values may be specified:

<code>gt</code>	The time of last access must be greater than the specified interval.
<code>eq</code>	The time of last access must be equal to the specified interval.
<code>gteq</code>	The time of last access must be greater than or equal to the specified interval.

For `<MAX>`, the following `Flags` attributes values may be specified.

<code>lt</code>	The time of last access must be less than the specified interval.
<code>lteq</code>	The time of last access must be less than or equal to the specified interval.

Including a `<MIN>` element in a `<WHEN>` clause causes VxFS to relocate files to which the rule applies that have been inactive for longer than the specified interval. Such a rule would typically be used to relocate inactive files to less expensive storage tiers. Conversely, including `<MAX>` causes files accessed within the specified interval to be relocated. It would typically be used to move inactive files against which activity had recommenced to higher performance or more reliable storage. Including both `<MIN>` and `<MAX>` causes VxFS to relocate files whose access age lies between the two.

The modification age relocation criterion, `<MODAGE>`, is similar to access age, except that files' POSIX `mtime` values are used in computations. You would typically specify the `<MODAGE>` criterion to cause relocation of recently modified files to higher performance or more reliable storage tiers in anticipation that the files would be accessed recurrently in the near future.

The file size relocation criterion, `<SIZE>`, causes files to be relocated if the files are larger or smaller than the values specified in the `<MIN>` and `<MAX>` relocation criteria, respectively, at the time that the `fsppadm enforce` command was issued. Specifying both criteria causes VxFS to schedule relocation for files whose sizes lie between the two. Using the `Units` attribute, threshold file sizes may be specified in the following units:

<code>bytes</code>	Bytes
<code>KB</code>	Kilobytes
<code>MB</code>	Megabytes

GB

Gigabytes

Specifying the I/O temperature relocation criterion

The I/O temperature relocation criterion, `<IOTEMP>`, causes files to be relocated if their I/O temperatures rise above or drop below specified values over a specified period immediately prior to the time at which the `fsppadm enforce` command was issued. A file's I/O temperature is a measure of the read, write, or total I/O activity against it normalized to the file's size. Higher I/O temperatures indicate higher levels of application activity; lower temperatures indicate lower levels. VxFS computes a file's I/O temperature by dividing the number of bytes transferred to or from it (read, written, or both) during the specified period by its size at the time that the `fsppadm enforce` command was issued.

See [“Calculating I/O temperature and access temperature”](#) on page 367.

As with the other file relocation criteria, `<IOTEMP>` may be specified with a lower threshold by using the `<MIN>` element, an upper threshold by using the `<MAX>` element, or as a range by using both. However, I/O temperature is dimensionless and therefore has no specification for units.

VxFS computes files' I/O temperatures over the period between the time when the `fsppadm enforce` command was issued and the number of days or hours in the past specified in the `<PERIOD>` element, where a day is a 24 hour period. The default unit of time is days. You can specify hours as the time unit by setting the `Units` attribute of the `<PERIOD>` element to `hours`. Symantec recommends that you specify hours only if you are using solid state disks (SSDs).

See [“Frequent scans”](#) on page 378.

For example, if you issued the `fsppadm enforce` command at 2 PM on Wednesday and you want VxFS to look at file I/O activity for the period between 2 PM on Monday and 2 PM on Wednesday, which is a period of 2 days, you would specify the following `<PERIOD>` element:

```
<PERIOD> 2 </PERIOD>
```

If you instead want VxFS to look at file I/O activity between 3 hours prior to running the `fsppadm enforce` command and the time that you ran the command, you specify the following `<PERIOD>` element:

```
<PERIOD Units="hours"> 3 </PERIOD>
```

The amount of time specified in the `<PERIOD>` element should not exceed one or two weeks due to the disk space used by the File Change Log (FCL) file.

I/O temperature is a softer measure of I/O activity than access age. With access age, a single access to a file resets the file's atime to the current time. In contrast, a file's I/O temperature decreases gradually as time passes without the file being accessed, and increases gradually as the file is accessed periodically. For example, if a new 10 megabyte file is read completely five times on Monday and `fsppadm enforce` runs at midnight, the file's two-day I/O temperature will be five and its access age in days will be zero. If the file is read once on Tuesday, the file's access age in days at midnight will be zero, and its two-day I/O temperature will have dropped to three. If the file is read once on Wednesday, the file's access age at midnight will still be zero, but its two-day I/O temperature will have dropped to one, as the influence of Monday's I/O will have disappeared.

If the intention of a file placement policy is to keep files in place, such as on top-tier storage devices, as long as the files are being accessed at all, then access age is the more appropriate relocation criterion. However, if the intention is to relocate files as the I/O load on them decreases, then I/O temperature is more appropriate.

The case for upward relocation is similar. If files that have been relocated to lower-tier storage devices due to infrequent access experience renewed application activity, then it may be appropriate to relocate those files to top-tier devices. A policy rule that uses access age with a low `<MAX>` value, that is, the interval between `fsppadm enforce` runs, as a relocation criterion will cause files to be relocated that have been accessed even once during the interval. Conversely, a policy that uses I/O temperature with a `<MIN>` value will only relocate files that have experienced a sustained level of activity over the period of interest.

Prefer attribute

You can specify a value for the `Prefer` attribute for the `<IOTEMP>` and `<ACCESSTEMP>` criteria, which gives preference to relocating files. The `Prefer` attribute can take two values: `low` or `high`. If you specify `low`, Veritas File System (VxFS) relocates the files with the lower I/O temperature before relocating the files with the higher I/O temperature. If you specify `high`, VxFS relocates the files with the higher I/O temperature before relocating the files with the lower I/O temperature. Symantec recommends that you specify a `Prefer` attribute value only if you are using solid state disks (SSDs).

See “[Prefer mechanism](#)” on page 377.

Different `<PERIOD>` elements may be used in the `<IOTEMP>` and `<ACCESSTEMP>` criteria of different `RELOCATE` statements within the same policy.

The following placement policy snippet gives an example of the `Prefer` criteria:

```
<RELOCATE>
```

```
...
```

```

<WHEN>
  <IOTEMP Type="nrbytes" Prefer="high">
    <MIN Flags="gteq"> 3.4 </MIN>
    <PERIOD Units="hours"> 6 </PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>

```

If there are a number of files whose I/O temperature is greater than the given minimum value, the files with the higher temperature are first subject to the `RELOCATE` operation before the files with the lower temperature.

Average I/O activity

The `Average` criteria allows you to specify the value of the I/O temperature as a ratio of per-file activity that occurs over the time specified by the `<PERIOD>` element compared to the overall file system activity that occurs over a longer period of time. The `<PERIOD>` element in the `RELOCATE` criteria specifies the a number of hours or days immediately before the time of the scan. During that time, the I/O statistics that are collected are used to process the files that are being scanned. Since I/O activity can change over time, collect the average I/O activity over a longer duration than the `<PERIOD>` value itself, which is by default 24 hours. Doing so lets you compute an average temperature of the whole file system. Symantec recommends that you specify an `Average` attribute value only if you are using solid state disks (SSDs).

See [“Average I/O activity”](#) on page 378.

The following placement policy snippet gives an example of the `Average` criteria:

```

<RELOCATE>
  ...
  <WHEN>
    <IOTEMP Type="nrbytes" Prefer="high" Average="*">
      <MIN Flags="gteq"> 1.5 </MIN>
      <PERIOD Units="hours"> 6 </PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>

```

In the snippet, VxFS relocates any file whose read IOTEMP over the last 6 hours is 1.5 times that of all the active files in the whole file system over the last 24 hours. This `Average` criteria is more intuitive and easier to specify than the absolute values.

The following formula computes the read IOTEMP of a given file:

$$\text{IOTEMP} = \frac{\text{(bytes of the file that are read in the PERIOD)}}{\text{(PERIOD in hours * size of the file in bytes)}}$$

The write and read/write IOTEMP are also computed accordingly.

The following formula computes the average read IOTEMP:

$$\text{Average IOTEMP} = \frac{\text{(bytes read of all active files in the last } h \text{ hours)}}{\text{(} h \text{ * size of all the active files in bytes)}}$$

h is 24 hours by default. The average write and read/write IOTEMP are also computed accordingly.

In the example snippet, the value 1.5 is the multiple of average read IOTEMP over the last 24 hours across the whole file system, or rather across all of the active inodes whose activity is still available in the File Change Log (FCL) file at the time of the scan. Thus, the files' read IOTEMP activity over the last 6 hours is compared against 1.5 times that of the last 24 hours average activity to make the relocation decision. Using this method eliminates the need to give a specific number for the `<IOTEMP>` or `<ACCESSTEMP>` criteria, and instead lets you specify a multiple of the Average temperature. Keeping this averaging period longer than the specified `<PERIOD>` value normalizes the effects of any spikes and lulls in the file activity.

You can also use the `Average` criteria with the `<ACCESSTEMP>` criteria. The purpose and usage are the same.

You determine the type of the average by whether you specify the `Average` criteria with the `<IOTEMP>` or with the `<ACCESSTEMP>` criteria. The `Average` criteria can be any of the following types, depending on the criteria used:

- read Average IOTEMP
- write Average IOTEMP
- rw Average IOTEMP
- read Average ACCESSTEMP
- write Average ACCESSTEMP
- rw Average ACCESSTEMP

The default `Average` is a 24 hour average temperature, which is the total of all of the temperatures available up to the last 24 hours in the FCL file, divided by the number of files for which such I/O statistics still exist in the FCL file. You can override the number of hours by specifying the `AveragePeriod` attribute in the `<PLACEMENT_POLICY>` element. Symantec recommends that you specify an `AveragePeriod` attribute value only if you are using solid state disks (SSDs).

The following example statement causes the average file system activity be collected and computed over a period of 30 hours instead of the default 24 hours:

```
<PLACEMENT_POLICY Name="Policy1" Version="5.1" AveragePeriod="30">
```

RELOCATE statement examples

The following example illustrates an unconditional relocation statement, which is the simplest form of the `RELOCATE` policy rule statement:

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
</RELOCATE>
```

The files designated by the rule's `SELECT` statement that reside on volumes in placement class `tier1` at the time the `fsppadm enforce` command executes would be unconditionally relocated to volumes in placement class `tier2` as long as space permitted. This type of rule might be used, for example, with applications that create and access new files but seldom access existing files once they have been processed. A `CREATE` statement would specify creation on `tier1` volumes, which are presumably high performance or high availability, or both. Each instantiation of `fsppadm enforce` would relocate files created since the last run to `tier2` volumes.

The following example illustrates a more comprehensive form of the `RELOCATE` statement that uses access age as the criterion for relocating files from `tier1` volumes to `tier2` volumes. This rule is designed to maintain free space on `tier1` volumes by relocating inactive files to `tier2` volumes:

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
```

```

    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gt">1</MIN>
      <MAX Flags="lt">1000</MAX>
    </SIZE>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>
  </WHEN>
</RELOCATE>

```

Files designated by the rule's `SELECT` statement are relocated from `tier1` volumes to `tier2` volumes if they are between 1 MB and 1000 MB in size and have not been accessed for 30 days. VxFS relocates qualifying files in the order in which it encounters them as it scans the file system's directory tree. VxFS stops scheduling qualifying files for relocation when when it calculates that already-scheduled relocations would result in `tier2` volumes being fully occupied.

The following example illustrates a possible companion rule that relocates files from `tier2` volumes to `tier1` ones based on their I/O temperatures. This rule might be used to return files that had been relocated to `tier2` volumes due to inactivity to `tier1` volumes when application activity against them increases. Using I/O temperature rather than access age as the relocation criterion reduces the chance of relocating files that are not actually being used frequently by applications. This rule does not cause files to be relocated unless there is sustained activity against them over the most recent two-day period.

```

<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier2</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes">

```

```

    <MIN Flags="gt">5</MIN>
    <PERIOD>2</PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>

```

This rule relocates files that reside on `tier2` volumes to `tier1` volumes if their I/O temperatures are above 5 for the two day period immediately preceding the issuing of the `fsppadm enforce` command. VxFS relocates qualifying files in the order in which it encounters them during its file system directory tree scan. When `tier1` volumes are fully occupied, VxFS stops scheduling qualifying files for relocation.

VxFS file placement policies are able to control file placement across any number of placement classes. The following example illustrates a rule for relocating files with low I/O temperatures from `tier1` volumes to `tier2` volumes, and to `tier3` volumes when `tier2` volumes are fully occupied:

```

<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier3</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes">
      <MAX Flags="lt">4</MAX>
      <PERIOD>3</PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>

```

This rule relocates files whose 3-day I/O temperatures are less than 4 and which reside on `tier1` volumes. When VxFS calculates that already-relocated files would result in `tier2` volumes being fully occupied, VxFS relocates qualifying files to

tier3 volumes instead. VxFS relocates qualifying files as it encounters them in its scan of the file system directory tree.

The `<FROM>` clause in the `RELOCATE` statement is optional. If the clause is not present, VxFS evaluates files designated by the rule's `SELECT` statement for relocation no matter which volumes they reside on when the `fspadm enforce` command is issued. The following example illustrates a fragment of a policy rule that relocates files according to their sizes, no matter where they reside when the `fspadm enforce` command is issued:

```
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MAX Flags="lt">10</MAX>
    </SIZE>
  </WHEN>
</RELOCATE>
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gteq">10</MIN>
      <MAX Flags="lt">100</MAX>
    </SIZE>
  </WHEN>
</RELOCATE>
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier3</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gteq">100</MIN>
```

```

    </SIZE>
  </WHEN>
</RELOCATE>

```

This rule relocates files smaller than 10 megabytes to `tier1` volumes, files between 10 and 100 megabytes to `tier2` volumes, and files larger than 100 megabytes to `tier3` volumes. VxFS relocates all qualifying files that do not already reside on volumes in their `DESTINATION` placement classes when the `fsppadm enforce` command is issued.

DELETE statement

The `DELETE` file placement policy rule statement is very similar to the `RELOCATE` statement in both form and function, lacking only the `<TO>` clause. File placement policy-based deletion may be thought of as relocation with a fixed destination.

Note: Use `DELETE` statements with caution.

The following XML snippet illustrates the general form of the `DELETE` statement:

```

<DELETE>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE>
      additional_placement_class_specifications
    </SOURCE>
  </FROM>
  <WHEN> relocation_conditions </WHEN>
</DELETE>

```

A `DELETE` statement contains the following clauses:

<code><FROM></code>	An optional clause that contains a list of placement classes from whose volumes designated files should be deleted if the files meet the conditions specified in the <code><WHEN></code> clause. No priority is associated with the ordering of placement classes in a <code><FROM></code> clause. If a file to which the rule applies is located on a volume in any specified placement class, the file is deleted. If a <code>DELETE</code> statement does not contain a <code><FROM></code> clause, VxFS deletes qualifying files no matter on which of a file system's volumes the files reside.
---------------------------	--

<WHEN> An optional clause specifying the conditions under which files to which the rule applies should be deleted. The form of the <WHEN> clause in a DELETE statement is identical to that of the <WHEN> clause in a RELOCATE statement. If a DELETE statement does not contain a <WHEN> clause, files designated by the rule's SELECT statement, and the <FROM> clause if it is present, are deleted unconditionally.

DELETE statement examples

The following example illustrates the use of the DELETE statement:

```
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS>tier3</CLASS>
    </SOURCE>
  </FROM>
</DELETE>
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS>tier2</CLASS>
    </SOURCE>
  </FROM>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">120</MIN>
    </ACCAGE>
  </WHEN>
</DELETE>
```

The first DELETE statement unconditionally deletes files designated by the rule's SELECT statement that reside on tier3 volumes when the `fspadm enforce` command is issued. The absence of a <WHEN> clause in the DELETE statement indicates that deletion of designated files is unconditional.

The second DELETE statement deletes files to which the rule applies that reside on tier2 volumes when the `fspadm enforce` command is issued and that have not been accessed for the past 120 days.

Calculating I/O temperature and access temperature

An important application of VxFS SmartTier is automating the relocation of inactive files to lower cost storage. If a file has not been accessed for the period of time specified in the `<ACCAGE>` element, a scan of the file system should schedule the file for relocation to a lower tier of storage. But, time since last access is inadequate as the only criterion for activity-based relocation.

Why time since last access is inadequate as the only criterion for activity-based relocation:

- Access age is a binary measure. The time since last access of a file is computed by subtracting the time at which the `fspadm enforce` command is issued from the POSIX `atime` in the file's metadata. If a file is opened the day before the `fspadm enforce` command, its time since last access is one day, even though it may have been inactive for the month preceding. If the intent of a policy rule is to relocate inactive files to lower tier volumes, it will perform badly against files that happen to be accessed, however casually, within the interval defined by the value of the `<ACCAGE>` parameter.
- Access age is a poor indicator of resumption of significant activity. Using `ACCAGE`, the time since last access, as a criterion for relocating inactive files to lower tier volumes may fail to schedule some relocations that should be performed, but at least this method results in less relocation activity than necessary. Using `ACCAGE` as a criterion for relocating previously inactive files that have become active is worse, because this method is likely to schedule relocation activity that is not warranted. If a policy rule's intent is to cause files that have experienced I/O activity in the recent past to be relocated to higher performing, perhaps more failure tolerant storage, `ACCAGE` is too coarse a filter. For example, in a rule specifying that files on `tier2` volumes that have been accessed within the last three days should be relocated to `tier1` volumes, no distinction is made between a file that was browsed by a single user and a file that actually was used intensively by applications.

SmartTier implements the concept of I/O temperature and access temperature to overcome these deficiencies. A file's I/O temperature is equal to the number of bytes transferred to or from it over a specified period of time divided by the size of the file. For example, if a file occupies one megabyte of storage at the time of an `fspadm enforce` operation and the data in the file has been completely read or written 15 times within the last three days, VxFS calculates its 3-day average I/O temperature to be 5 (15 MB of I/O ÷ 1 MB file size ÷ 3 days).

Similarly, a file's average access temperature is the number of read or write requests made to it over a specified number of 24-hour periods divided by the number of periods. Unlike I/O temperature, access temperature is unrelated to

file size. A large file to which 20 I/O requests are made over a 2-day period has the same average access temperature as a small file accessed 20 times over a 2-day period.

If a file system's active placement policy includes any `<IOTEMP>` or `<ACCESSTEMP>` clauses, VxFS begins policy enforcement by using information in the file system's FCL file to calculate average I/O activity against all files in the file system during the longest `<PERIOD>` specified in the policy. Shorter specified periods are ignored. VxFS uses these calculations to qualify files for I/O temperature-based relocation and deletion.

Note: If FCL is turned off, I/O temperature-based relocation will not be accurate. When you invoke the `fsppadm enforce` command, the command displays a warning if the FCL is turned off.

As its name implies, the File Change Log records information about changes made to files in a VxFS file system. In addition to recording creations, deletions, extensions, the FCL periodically captures the cumulative amount of I/O activity (number of bytes read and written) on a file-by-file basis. File I/O activity is recorded in the FCL each time a file is opened or closed, as well as at timed intervals to capture information about files that remain open for long periods.

If a file system's active file placement policy contains `<IOTEMP>` clauses, execution of the `fsppadm enforce` command begins with a scan of the FCL to extract I/O activity information over the period of interest for the policy. The period of interest is the interval between the time at which the `fsppadm enforce` command was issued and that time minus the largest interval value specified in any `<PERIOD>` element in the active policy.

For files with I/O activity during the largest interval, VxFS computes an approximation of the amount of read, write, and total data transfer (the sum of the two) activity by subtracting the I/O levels in the oldest FCL record that pertains to the file from those in the newest. It then computes each file's I/O temperature by dividing its I/O activity by its size at `Tscan`. Dividing by file size is an implicit acknowledgement that relocating larger files consumes more I/O resources than relocating smaller ones. Using this algorithm requires that larger files must have more activity against them in order to reach a given I/O temperature, and thereby justify the resource cost of relocation.

While this computation is an approximation in several ways, it represents an easy to compute, and more importantly, unbiased estimate of relative recent I/O activity upon which reasonable relocation decisions can be based.

File relocation and deletion decisions can be based on read, write, or total I/O activity.

The following XML snippet illustrates the use of `IOTEMP` in a policy rule to specify relocation of low activity files from `tier1` volumes to `tier2` volumes:

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrwbytes">
      <MAX Flags="lt">3</MAX>
      <PERIOD Units="days">4</PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>
```

This snippet specifies that files to which the rule applies should be relocated from `tier1` volumes to `tier2` volumes if their I/O temperatures fall below 3 over a period of 4 days. The `Type="nrwbytes"` XML attribute specifies that total data transfer activity, which is the sum of bytes read and bytes written, should be used in the computation. For example, a 50 megabyte file that experienced less than 150 megabytes of data transfer over the 4-day period immediately preceding the `fsppadm enforce` scan would be a candidate for relocation. VxFS considers files that experience no activity over the period of interest to have an I/O temperature of zero. VxFS relocates qualifying files in the order in which it encounters the files in its scan of the file system directory tree.

Using I/O temperature or access temperature rather than a binary indication of activity, such as the POSIX `atime` or `mtime`, minimizes the chance of not relocating files that were only accessed occasionally during the period of interest. A large file that has had only a few bytes transferred to or from it would have a low I/O temperature, and would therefore be a candidate for relocation to `tier2` volumes, even if the activity was very recent.

But, the greater value of I/O temperature or access temperature as a file relocation criterion lies in upward relocation: detecting increasing levels of I/O activity against files that had previously been relocated to lower tiers in a storage hierarchy

due to inactivity or low temperatures, and relocating them to higher tiers in the storage hierarchy.

The following XML snippet illustrates relocating files from `tier2` volumes to `tier1` when the activity level against them increases.

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier2</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes">
      <MAX Flags="gt">5</MAX>
      <PERIOD Units="days">2</PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>
```

The `<RELOCATE>` statement specifies that files on `tier2` volumes whose I/O temperature as calculated using the number of bytes read is above 5 over a 2-day period are to be relocated to `tier1` volumes. Bytes written to the file during the period of interest are not part of this calculation.

Using I/O temperature rather than a binary indicator of activity as a criterion for file relocation gives administrators a granular level of control over automated file relocation that can be used to attune policies to application requirements. For example, specifying a large value in the `<PERIOD>` element of an upward relocation statement prevents files from being relocated unless I/O activity against them is sustained. Alternatively, specifying a high temperature and a short period tends to relocate files based on short-term intensity of I/O activity against them.

I/O temperature and access temperature utilize the `sqlite3` database for building a temporary table indexed on an inode. This temporary table is used to filter files based on I/O temperature and access temperature. The temporary table is stored in the database file `._fsppadm_fcliotemp.db`, which resides in the `lost+found` directory of the mount point.

Multiple criteria in file placement policy rule statements

In certain cases, file placement policy rule statements may contain multiple clauses that affect their behavior. In general, when a rule statement contains multiple clauses of a given type, all clauses must be satisfied in order for the statement to be effective. There are four cases of note in which multiple clauses may be used.

Multiple file selection criteria in SELECT statement clauses

Within a single `SELECT` statement, all the selection criteria clauses of a single type are treated as a selection list. A file need only satisfy a single criterion of a given type to be designated.

In the following example, files in any of the `db/datafiles`, `db/indexes`, and `db/logs` directories, all relative to the file system mount point, would be selected:

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
</SELECT>
```

This example is in direct contrast to the treatment of selection criteria clauses of different types. When a `SELECT` statement includes multiple types of file selection criteria, a file must satisfy one criterion of each type in order for the rule's action statements to apply.

In the following example, a file must reside in one of `db/datafiles`, `db/indexes`, or `db/logs` and be owned by one of `DBA_Manager`, `MFG_DBA`, or `HR_DBA` to be designated for possible action:

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
  <USER>DBA_Manager</USER>
  <USER>MFG_DBA</USER>
  <USER>HR_DBA</USER>
</SELECT>
```

If a rule includes multiple `SELECT` statements, a file need only satisfy one of them to be selected for action. This property can be used to specify alternative conditions for file selection.

In the following example, a file need only reside in one of db/datafiles, db/indexes, or db/logs or be owned by one of DBA_Manager, MFG_DBA, or HR_DBA to be designated for possible action:

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
</SELECT>
<SELECT>
  <USER>DBA_Manager</USER>
  <USER>MFG_DBA</USER>
  <USER>HR_DBA</USER>
</SELECT>
```

Multiple placement classes in <ON> clauses of CREATE statements and in <TO> clauses of RELOCATE statements

Both the <ON> clause of the CREATE statement and the <TO> clause of the RELOCATE statement can specify priority ordered lists of placement classes using multiple <DESTINATION> XML elements. VxFS uses a volume in the first placement class in a list for the designated purpose of file creation or relocation, if possible. If no volume in the first listed class has sufficient free space or if the file system's volume set does not contain any volumes with that placement class, VxFS uses a volume in the second listed class if possible. If no volume in the second listed class can be used, a volume in the third listed class is used if possible, and so forth.

The following example illustrates of three placement classes specified in the <ON> clause of a CREATE statement:

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier3</CLASS>
    </DESTINATION>
  </ON>
</CREATE>
```

In this statement, VxFS would allocate space for newly created files designated by the rule's `SELECT` statement on `tier1` volumes if space was available. If no `tier1` volume had sufficient free space, VxFS would attempt to allocate space on a `tier2` volume. If no `tier2` volume had sufficient free space, VxFS would attempt allocation on a `tier3` volume. If sufficient space could not be allocated on a volume in any of the three specified placement classes, allocation would fail with an `ENOSPC` error, even if the file system's volume set included volumes in other placement classes that did have sufficient space.

The `<TO>` clause in the `RELOCATE` statement behaves similarly. VxFS relocates qualifying files to volumes in the first placement class specified if possible, to volumes in the second specified class if not, and so forth. If none of the destination criteria can be met, such as if all specified classes are fully occupied, qualifying files are not relocated, but no error is signaled in this case.

Multiple placement classes in `<FROM>` clauses of `RELOCATE` and `DELETE` statements

The `<FROM>` clause in `RELOCATE` and `DELETE` statements can include multiple source placement classes. However, unlike the `<ON>` and `<TO>` clauses, no order or priority is implied in `<FROM>` clauses. If a qualifying file resides on a volume in any of the placement classes specified in a `<FROM>` clause, it is relocated or deleted regardless of the position of its placement class in the `<FROM>` clause list of classes.

Multiple conditions in `<WHEN>` clauses of `RELOCATE` and `DELETE` statements

The `<WHEN>` clause in `RELOCATE` and `DELETE` statements may include multiple relocation criteria. Any or all of `<ACCAGE>`, `<MODAGE>`, `<SIZE>`, and `<IOTEMP>` can be specified. When multiple conditions are specified, all must be satisfied in order for a selected file to qualify for relocation or deletion.

In the following example, a selected file would have to be both inactive, that is, not accessed, for more than 30 days and larger than 100 megabytes to be eligible for relocation or deletion:

```
<WHEN>
  <ACCAGE Units="days">
    <MIN Flags="gt">30</MIN>
  </ACCAGE>
  <SIZE Units="MB">
    <MIN Flags="gt">100</MIN>
```

```
</SIZE>  
</WHEN>
```

You cannot write rules to relocate or delete a single designated set of files if the files meet one of two or more relocation or deletion criteria.

File placement policy rule and statement ordering

You can use the SmartTier graphical user interface (GUI) to create any of four types of file placement policy documents. Alternatively, you can use a text editor or XML editor to create XML policy documents directly. The GUI places policy rule statements in the correct order to achieve the desired behavior. If you use a text editor, it is your responsibility to order policy rules and the statements in them so that the desired behavior results.

The rules that comprise a placement policy may occur in any order, but during both file allocation and `fsppadm enforce` relocation scans, the first rule in which a file is designated by a `SELECT` statement is the only rule against which that file is evaluated. Thus, rules whose purpose is to supersede a generally applicable behavior for a special class of files should precede the general rules in a file placement policy document.

The following XML snippet illustrates faulty rule placement with potentially unintended consequences:

```
<?xml version="1.0"?>  
<!DOCTYPE FILE_PLACEMENT_POLICY SYSTEM "placement.dtd">  
<FILE_PLACEMENT_POLICY Version="5.0">  
  <RULE Name="GeneralRule">  
    <SELECT>  
      <PATTERN>*</PATTERN>  
    </SELECT>  
    <CREATE>  
      <ON>  
        <DESTINATION>  
          <CLASS>tier2</CLASS>  
        </DESTINATION>  
      </ON>  
    </CREATE>  
    other_statements  
  </RULE>  
  <RULE Name="DatabaseRule">  
    <SELECT>  
      <PATTERN>*.db</PATTERN>
```

```

</SELECT>
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </ON>
</CREATE>
  other_statements
</RULE>
</FILE_PLACEMENT_POLICY>

```

The `GeneralRule` rule specifies that all files created in the file system, designated by `<PATTERN>*</PATTERN>`, should be created on `tier2` volumes. The `DatabaseRule` rule specifies that files whose names include an extension of `.db` should be created on `tier1` volumes. The `GeneralRule` rule applies to any file created in the file system, including those with a naming pattern of `*.db`, so the `DatabaseRule` rule will never apply to any file. This fault can be remedied by exchanging the order of the two rules. If the `DatabaseRule` rule occurs first in the policy document, VxFS encounters it first when determining where to new place files whose names follow the pattern `*.db`, and correctly allocates space for them on `tier1` volumes. For files to which the `DatabaseRule` rule does not apply, VxFS continues scanning the policy and allocates space according to the specification in the `CREATE` statement of the `GeneralRule` rule.

A similar consideration applies to statements within a placement policy rule. VxFS processes these statements in order, and stops processing on behalf of a file when it encounters a statement that pertains to the file. This can result in unintended behavior.

The following XML snippet illustrates a `RELOCATE` statement and a `DELETE` statement in a rule that is intended to relocate if the files have not been accessed in 30 days, and delete the files if they have not been accessed in 90 days:

```

<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>

```

```
</WHEN>  
</RELOCATE>  
<DELETE>  
  <WHEN>  
    <ACCAGE Units="days">  
      <MIN Flags="gt">90</MIN>  
    </ACCAGE>  
  </WHEN>  
</DELETE>
```

As written with the `RELOCATE` statement preceding the `DELETE` statement, files will never be deleted, because the `<WHEN>` clause in the `RELOCATE` statement applies to all selected files that have not been accessed for at least 30 days. This includes those that have not been accessed for 90 days. VxFS ceases to process a file against a placement policy when it identifies a statement that applies to that file, so the `DELETE` statement would never occur. This example illustrates the general point that `RELOCATE` and `DELETE` statements that specify less inclusive criteria should precede statements that specify more inclusive criteria in a file placement policy document. The GUI automatically produce the correct statement order for the policies it creates.

File placement policies and extending files

In a VxFS file system with an active file placement policy, the placement class on whose volume a file resides is part of its metadata, and is attached when it is created and updated when it is relocated. When an application extends a file, VxFS allocates the incremental space on the volume occupied by the file if possible. If not possible, VxFS allocates the space on another volume in the same placement class. For example, if a file is created on a `tier1` volume and later relocated to a `tier2` volume, extensions to the file that occur before the relocation have space allocated on a `tier1` volume, while those occurring after to the relocation have their space allocated on `tier2` volumes. When a file is relocated, all of its allocated space, including the space acquired by extension, is relocated to `tier2` volumes in this case.

Using SmartTier with solid state disks

The SmartTier feature has been enhanced with the following placement policy features to support SSD-based tiers:

- Allowance of fine grained temperatures, such as allowing hours as units for the `<IOTEMP>` and `<ACCESSTEMP>` criteria

- Support of the `Prefer` attribute for the `<IOTEMP>` and `<ACCESSTEMP>` criteria
- Provision of a mechanism to relocate based on average I/O activity
- Reduction of the intensity and duration of scans to minimize the impact on resources, such as memory, CPU, and I/O bandwidth
- Quick identification of cold files

To gain these benefits, you must modify the existing placement policy as per the latest version of the DTD and assign the policy again. However, existing placement policies continue to function as before. You do not need to update the placement policies if you do not use the new features.

Fine grain temperatures

Before the solid state disk (SSD) enhancements, the SmartTier feature computed temperature values on a day granularity. Day granularity is the I/O activity per day over at least one day. As such, the `<PERIOD>` element had to be in days for the `<IOTEMP>` and `<ACCESSTEMP>` criteria. With SSDs, relocation decisions might need to happen within the day itself, based on I/O activity that Veritas File System (VxFS) measured over a shorter duration. As such, you can now specify "hours" for the `Units` attribute value for the `<IOTEMP>` and `<ACCESSTEMP>` criteria.

See [“Specifying the I/O temperature relocation criterion”](#) on page 357.

The following placement policy snippet gives an example of specifying 4 hours as the period of time:

```
<RELOCATE>
...
<WHEN>
  <IOTEMP Type="nwbytes">
    <MIN Flags="gteq"> 2 </MIN>
    <PERIOD Units="hours"> 4 </PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>
```

Prefer mechanism

You can now specify a value for the `Prefer` attribute for the `<IOTEMP>` and `<ACCESSTEMP>` criteria, which gives preference to relocating files.

See [“Prefer attribute”](#) on page 358.

In case of a solid state disk (SSD)-based tier, you might want to relocate a file to an SSD as soon as there is a marked increase in the I/O activity. However, once

Veritas File System (VxFS) has relocated the file to an SSD, it may be beneficial to keep the file on the SSD as long as the activity remains high to avoid frequent thrashing. You want to watch the activity for some time longer than the time that you watched the activity when you relocated the file to the SSD before you decide to move the file off of the SSD.

The following placement policy snippet gives an example of the `Prefer` criteria:

```
<RELOCATE>
...
<WHEN>
  <IOTEMP Type="nrbytes" Prefer="high">
    <MIN Flags="gteq"> 3.4 </MIN>
    <PERIOD Units="hours"> 6 </PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>
```

If there are a number of files whose I/O temperature is greater than the given minimum value, the files with the higher temperature are first subject to the `RELOCATE` operation before the files with the lower temperature. This is particularly useful in case of SSDs, which are limited in size and are expensive. As such, you generally want to use SSDs for the most active files.

Average I/O activity

Before the solid state disk (SSD) enhancements, you were required to specify an absolute value of the temperature when you used the `ACCESSTEMP` criteria and `IOTEMP` criteria in the SmartTier placement policies. However, arriving at such absolute numbers is difficult and requires you to experiment and observe data access patterns over a period of time. Moreover, over a period of time, you might have to change this value due to changing access patterns. As such, you might need to repeat the experiment. To ease constructing `ACCESSTEMP` and `IOTEMP`-based policies, a new criteria has been introduced: `Average`.

See [“Average I/O activity”](#) on page 359.

Frequent scans

You now can specify "hours" for the `Units` attribute value, and as such the I/O stats collection `PERIOD` can be much shorter than in previous releases. When not using solid state disks (SSDs), you can only specify "days" for the `Units` attribute value, which might be sufficient for your needs. However, a `PERIOD` shorter than a day is required in the context of using SSDs since the candidate files and their

activity levels can change during the day. As a result, SmartTier must scan more frequently, which leads to a higher scan load on the host systems.

You must satisfy the following conflicting requirements simultaneously:

- Bring down the temperature collection windows to hourly levels.
- Reduce the impact of more frequent scans on resources, such as CPU, I/O, and memory.

The following scheme is an example of one way to reduce the impact of frequent scans:

- Confine the scan to only active files during the PERIOD by focusing only on the files that showed any activity in the File Change Log (FCL) by running the `fspadm` command with the `-C` option.
See [“Quick identification of cold files”](#) on page 379.
- Scan frequently, such as every few hours. Frequent scans potentially reduce the number of inodes that VxFS touches and logs in the File Change Log (FCL) file, thereby limiting the duration of each scan. As such, the changes that VxFS collects in the FCL file since the last scan provide details on fewer active files.
- Use the `<IOTEMP>` and `<ACCESSTEMP>` criteria to promote files to SSDs more aggressively, which leaves cold files sitting in SSDs.

Quick identification of cold files

The placement mechanism generally leaves the cold files in solid state disks (SSDs) if the files continue to remain inactive. This results in a lack of room for active files if the active files need to be moved into SSDs, and thus results in ineffective use of storage. An SSD enhancement for identifying cold files quickly solves this problem.

The enhancement is a method for quickly identifying files on a particular tier of the SmartTier file system so that the files can be relocated if necessary. The method consists of a map that associates storage devices with the inodes of files residing on the storage devices.

Veritas File System (VxFS) updates the file location map during the following times:

- SmartTier’s own file relocations
- On examination of the file system’s File Change Log (FCL) for changes that are made outside of SmartTier’s scope.

Both of these updates occur during SmartTier’s relocation scans, which are typically scheduled to occur periodically. But, you can also update the file location map anytime by running the `fspadm` command with the `-T` option.

The `-c` option is useful to process active files before any other files. For best results, specify the `-T` option in conjunction with the `-c` option. Specifying both the `-T` option and `-c` option causes the `fssppadm` command to evacuate any cold files first to create room in the SSD tier to accommodate any active files that will be moved into the SSD tier via the `-c` option. Specifying `-c` in conjunction with `-T` confines the scope of the scan, which consumes less time and resources, and thus allows frequent scans to meet the dynamic needs of data placement.

See [“Enforcing a placement policy”](#) on page 342.

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

With the help of the `map`, instead of scanning the full file system, you can confine the scan to only the files on the SSD tiers in addition to the active files that VxFS recorded in the FCL. This scheme potentially achieves the dual purpose of reducing the temperature time granularity and at the same time reducing the scan load.

Example placement policy when using solid state disks

The following snippet is one possible placement policy for use with solid state disk (SSD)-based tiers.

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfss/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="SSD_policy">
  <RULE Flags="data" Name="all_files">
    <COMMENT>
      The first two RELOCATES will do the evacuation
      out of SSDs to create room for any relocations
      into the SSDs by the third RELOCATE. The parameters
      that can be tuned are basically values for PERIOD and
      the values of MIN and/or MAX as the per the case.
      The values for MIN and MAX are treated as multiples of
      average activity over past 24 hour period.
    </COMMENT>
    <SELECT>
      <PATTERN> * </PATTERN>
    </SELECT>

    <CREATE>
      <COMMENT>
        create files on ssdtier, failing which
        create them on other tiers
      </COMMENT>
    <ON>
```

```
<DESTINATION Flags="any">
  <CLASS> ssdtier </CLASS>
</DESTINATION>
</ON>
</CREATE>

<RELOCATE>
  <COMMENT>
    Move the files out of SSD if their last 3 hour
    write IOTEMP is more than 1.5 times the last
    24 hour average write IOTEMP. The PERIOD is
    purposely shorter than the other RELOCATEs
    because we want to move it out as soon as
    write activity starts peaking. This criteria
    could be used to reduce SSD wear outs.
  </COMMENT>
  <FROM>
    <SOURCE>
      <CLASS> ssdtier </CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS> nonssd_tier </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nwbytes" Average="*">
      <MIN Flags="gt"> 1.5 </MIN>
      <PERIOD Units="hours"> 3 </PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>

<RELOCATE>
  <COMMENT>
    OR move the files out of SSD if their last 6 hour
    read IOTEMP is less than half the last 24 hour
    average read IOTEMP. The PERIOD is longer,
    we may want to observe longer periods
    having brought the file in. This avoids quickly
    sending the file out of SSDs once in.
  </COMMENT>
```

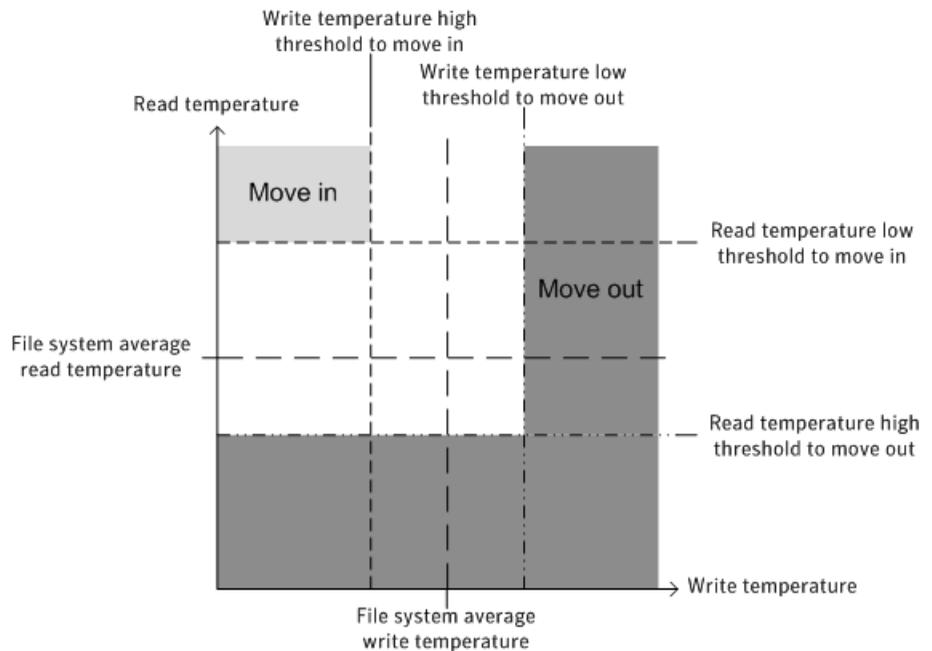
```
<FROM>
  <SOURCE>
    <CLASS> ssdtier </CLASS>
  </SOURCE>
</FROM>
<TO>
  <DESTINATION>
    <CLASS> nonssd_tier </CLASS>
  </DESTINATION>
</TO>
<WHEN>
  <IOTEMP Type="nrbytes" Average="*">
    <MAX Flags="lt"> 0.5 </MAX>
    <PERIOD Units="hours"> 6 </PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>

<RELOCATE>
  <COMMENT>
    OR move the files into SSD if their last 3 hour
    read IOTEMP is more than or equal to 1.5 times
    the last 24 hour average read IOTEMP AND
    their last 6 hour write IOTEMP is less than
    half of the last 24 hour average write IOTEMP
  </COMMENT>
<TO>
  <DESTINATION>
    <CLASS> ssd_tier </CLASS>
  </DESTINATION>
</TO>
<WHEN>
  <IOTEMP Type="nrbytes" Prefer="high" Average="*">
    <MIN Flags="gteq"> 1.5 </MIN>
    <PERIOD Units="hours"> 3 </PERIOD>
  </IOTEMP>
  <IOTEMP Type="nwbytes" Average="*">
    <MAX Flags="lt"> 0.5 </MAX>
    <PERIOD Units="hours"> 3 </PERIOD>
  </IOTEMP>
</WHEN>
</RELOCATE>
```

```
</RULE>
</PLACEMENT_POLICY>
```

In this placement policy, new files are created on the SSD tiers if space is available, or elsewhere if space is not available. When enforce is performed, the files that are currently in SSDs whose write activity is increased above a threshold or whose read activity fell below a threshold over a given period are moved out of the SSDs. The first two RELOCATES capture this intent. However, the files whose read activity intensified above a threshold and whose write activity does not exceed a threshold over the given period are moved into SSDs, while giving preference to files with higher read activity.

The following figure illustrates the behavior of the example placement policy:



The files whose I/O activity falls in the light gray area are good candidates for moving in to SSD storage. These files have less write activity such that they have less impact on wear leveling, and the slower write times to SSDs is less of a factor. These files have intense read activity, which also makes the files ideal for placement on SSDs since read activity does not cause any wear leveling side effects, and reads are faster from SSDs. In contrast, the files whose I/O activity falls in the dark gray area are good candidates to be moved out of SSD storage, since they have more write activity or less read activity. Greater write activity leads to greater wear leveling of the SSDs, and your file system's performance suffers from the

slower write times of SSDs. Lesser read activity means that you are not benefitting from the faster read times of SSDs with these files.

Migrating data

- [Chapter 26. Understanding data migration](#)
- [Chapter 27. Offline data migration](#)
- [Chapter 28. Online migration of a native file system to the VxFS file system](#)
- [Chapter 29. Migrating data between platforms](#)

Understanding data migration

This chapter includes the following topics:

- [Types of data migration](#)

Types of data migration

This section describes the following types of data migration:

- **Migrating data from LVM to Storage Foundation using offline migration**
When you install Storage Foundation, you may already have some volumes that are controlled by the Logical Volume Manager. You can preserve your data and convert these volumes to Veritas Volume Manager volumes. See [“About VxVM and LVM”](#) on page 389.
- **Migrating data from a native file system to a Veritas File System (VxFS) file system using online migration**
- **Migrating data between platforms using Cross-platform Data Sharing (CDS)**
Storage Foundation lets you create disks and volumes so that the data can be read by systems running different operating systems. CDS disks and volumes cannot be mounted and accessed from different operating systems at the same time. The CDS functionality provides an easy way to migrate data between one system and another system running a different operating system. See [“Overview of CDS”](#) on page 451.
- **Migrating data between arrays**
Storage Foundation supports arrays from various vendors. If your storage needs change, you can move your data between arrays.

Note: The procedures are different if you plan to migrate to a thin array from a thick array.

See [“Migrating to thin provisioning”](#) on page 114.

Offline data migration

This chapter includes the following topics:

- [About VxVM and LVM](#)
- [Converting LVM, JFS and JFS2 to VxVM and VxFS](#)
- [Command differences](#)
- [System Management Interface Tool \(SMIT\)](#)

About VxVM and LVM

This section includes a brief description of the benefits of migrating from the AIX Logical Volume Manager (LVM) to VxVM (including migration of JFS or JFS2 file systems to the Veritas File System, VxFS), and the coexistence of VxVM disks with LVM disks is also given.

About Veritas Volume Manager

This section provides an overview of Veritas Volume Manager by Symantec (also referred to as VxVM) and its features.

Veritas Volume Manager is an alternative Volume Management product for AIX that includes mirroring features. It offers many capabilities that are not available with the AIX LVM products today.

Veritas Volume Manager can coexist with LVM. Users can decide which volumes they want managed by each volume manager. For users who want to migrate LVM volume groups to VxVM disk groups, a conversion utility, `vxconvert`, is included.

See “[About converting LVM, JFS and JFS2 configurations](#)” on page 395.

Users may choose to use Veritas Volume Manager for their non-root disks.

Notable features of VxVM

Veritas Volume Manager provides many features, some of which are not available with LVM. Notable VxVM features are described in the list below. See the other Veritas Volume Manager documents for more details about using these features.

Some features in VxVM are not available under LVM.

See [“Tasks with no direct LVM equivalents”](#) on page 434.

Veritas Volume Manager includes the following features:

- Concatenation, the combining of discontinuous disk regions into virtual devices.
- Spanning, concatenation across different physical media.
- Striping, distribution of storage mappings for a virtual device so that multi-threaded accesses tend to cause even use of all physical media.
- Dynamic Multipathing (DMP) for Active/Active and Active/Passive devices. DMP provides higher availability to data on disks with multiple host-to-device pathways by providing a disk/device path failover mechanism. In the event of a loss of one connection to a disk, the system continues to access the data over the other available connections to the disk. DMP also provides in some cases, improved I/O performance from disks with multiple concurrently available pathways by balancing the I/O load uniformly across multiple I/O paths to the disk device. LVM supports path failover but does not support I/O balancing. DMP support may be used with devices that show improved performance when I/O is balanced across the multiple paths such as IBM SHARK, EMC Symmetrix disk array, and other OEM array devices.
- Free Space Management, providing simple goal-based allocation of storage.
- Task Monitor, which tracks the progress of system recovery by monitoring task creation, maintenance, and completion. The Task Monitor allows you to pause, resume, and stop as desired to adjust the impact on system performance.
- Support for VxFS (not available with LVM).

The following Veritas Volume Manager features may require an additional license:

- Multiple mirroring with up to 32 mirror copies of a volume’s address space.
- Mirrored stripes, enabling mirroring of individual data stripes to spread data across multiple disks while providing data redundancy. This layout dramatically increases the ability to handle multiple disk failures and reduces the amount of resynchronization needed. If a disk fails, the data on the surviving disks can be used to reconstruct and recover the lost data.
- Hot-relocation, which allows a system to react automatically to I/O failures on redundant (mirrored or RAID-5) VxVM objects, restoring redundancy and access to those objects without administrative intervention. VxVM detects I/O

failures on objects and relocates the affected subdisks. The `vxunreloc` utility can be used to restore the system to the same configuration that existed before the disk failure.

- RAID-5, which provides data redundancy by using parity, at a lower storage cost than mirroring. RAID-5 provides data redundancy by using parity. Parity is a calculated value used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is calculated by doing an exclusive OR (XOR) procedure on the data. The resulting parity is then written in an interleaved fashion to the RAID-5 array established by the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and parity information.
- Online Data Migration, which allows for regions of storage on physical media to be dynamically moved to other physical devices.
- Online Relayout or Dynamic Restriping, the ability to change logical data configuration while online, for example, to change RAID-5 to a mirrored layout or to change a stripe unit size. The volume data remains available during the relayout.
- Striped mirrors (RAID-0 + RAID-1), which provide a layered volume structure that tolerates failure better and which has greater redundancy than a mirrored stripe (RAID-1 + RAID-0) structure. Each subdisk is mirrored so that recovery is quicker (only the subdisk is recovered instead of a full mirror).
See the *Veritas Volume Manager Administrator's Guide*.
- Improved RAID-5 subdisk, using layered volume technology where the RAID-5 subdisk move operation leaves the old subdisk in place while the new one is being synchronized, thus maintaining redundancy and resiliency to failures during the move.
- Veritas FlashSnap™ which includes the disk group split and join, and Persistent FastResync features of Veritas Volume Manager as well as the Storage Checkpoint features of Veritas File System.
See the *Veritas FlashSnap Point-In-Time Copy Solutions Administrator's Guide*.

For more information on LVM, refer to *AIX Logical Volume Manager, from A to Z: Introduction and Concepts*.

For information on VxVM commands, refer to the *Veritas Volume Manager Administrator's Guide*.

A full list of the many new features that are supported in VxVM 5.1 SP1 is given in the *Release Notes*.

VxVM and LVM, a conceptual comparison

The following section compares the terminology used in LVM and VxVM at a conceptual level.

For more information, refer to the glossary of this Guide for precise and detailed definitions of these terms.

Disk storage management

LVM term: LVM

VxVM term: VxVM

Both LVM and VxVM enable online disk storage management. They both build virtual devices, called volumes, on physical disks. Volumes are not limited by the underlying physical disks, and can include other virtual objects such as mirrors. Volumes are accessed through the AIX file system, a database, or other applications in the same manner as physical disks would be accessed.

Physical volumes and disks

LVM term: Physical volume

VxVM term: VxVM disk

An LVM physical volume and a VxVM disk are conceptually the same. A physical disk is the basic storage device (media) where the data is ultimately stored. You can access the data on a physical disk by using a device name (`devname`) to locate the disk.

In LVM, a disk that has been initialized by LVM becomes known as a physical volume.

A VxVM disk is one that is placed under the Volume Manager control and is added to a disk group.

VxVM can place a disk under its control without adding it to a disk group. The VxVM Storage Administrator shows these disks as “free space pool”.

Volumes

LVM term: Logical volume

VxVM term: Volume

An LVM logical volume and a VxVM volume are conceptually the same. Both are virtual disk devices that appear to applications, databases, and file systems like physical disk devices, but do not have the physical limitations of physical disk

devices. Due to its virtual nature, a volume (LVM or VxVM) is not restricted to a particular disk or a specific area of a disk.

An LVM volume is composed of fixed length extents. LVM volumes can be mirrored or striped.

VxVM volumes consist of one or more plexes/mirrors holding a copy of the data in the volume which in turn are made up of subdisks with arbitrary length. The configuration of a volume can be changed by using the VxVM user interfaces.

VxVM volumes can be one of four types: mirrored, RAID-5, striped, or concatenated.

See the *Veritas Volume Manager Administrator's Guide*.

Groups

LVM term: Volume group

VxVM term: Disk group

LVM volume groups are conceptually similar to VxVM disk groups.

An LVM volume group is the collective identity of a set of physical volumes, which provide disk storage for the logical volumes.

A VxVM disk group is a collection of VxVM disks that share a common configuration. A configuration is a set of records with detailed information about related VxVM objects, their attributes, and their associations.

In addition, both LVM and VxVM have the following characteristics:

Volumes can be mapped to multiple VxVM disks or LVM physical volumes.

VxVM disks must reside in only one disk group, and LVM physical volumes must reside in one volume group.

Physical extents and subdisks

LVM term: Physical extent

VxVM term: Subdisk

User data is contained in physical extents in LVM and subdisks in VxVM.

The LVM physical extents are of a fixed length. LVM allocates space in terms of physical extents which is a set of physical disk blocks on a physical volume. The extent size for all physical volumes within a volume group must be the same.

VxVM allocates disk space in term of subdisks which is a set of physical disk blocks representing a specific portion of a VxVM disk and is of arbitrary size.

VGRA and the private region

LVM term: VGRA

VxVM term: Private region

VGRA and the Private Region are similar conceptually.

In LVM, the VGRA is the region in the disk which stores metadata.

In VxVM, the private region of a disk contains various on-disk structures that are used by the Volume Manager for various internal purposes. Private regions can also contain copies of a disk group's configuration, and copies of the disk group's kernel log.

Free space

LVM term: Unused physical extent

VxVM term: Free space

LVM contains unused physical extents that are not part of a logical volume, but are part of the volume group.

Similarly, free space is an area of a disk under VxVM that is not allocated to any subdisk or reserved for use by any other Volume Manager object.

Mirrors

LVM term: Mirrors

VxVM term: Mirrors (plexes)

Both LVM and VxVM support mirrors. Mirrors can be used to store multiple copies of a volume's data on separate disks.

Mirrors allow duplicate copies of the extents to be kept on separate physical volumes. AIX LVM supports up to 3 mirrors.

A VxVM mirror consists of plexes. Each plex is a copy of the volume. A plex consists of one or more subdisks located on one or more disks. VxVM volumes can have up to 32 mirrors (where each plex is a copy of data). Mirroring features are available with an additional license.

Export and deport

LVM term: Export

VxVM term: Deport

In LVM, exporting removes volume group information. The volume group must have already been deactivated.

Similarly in VxVM, `deport` makes a disk group inaccessible by the system.

Import

LVM term: Import

VxVM term: Import

In LVM, `import` adds a volume group to the system and the volume group information to `/etc/filesystems` but does not make the volumes accessible. The volume group must be activated by the `varyonvg` command in order to make volumes accessible.

In VxVM, `vxpdg import` imports a disk group and makes the disk group accessible by the system.

Bad block pool

LVM term: Bad block pool

VxVM term: No similar term

In LVM, the bad block pool provides for the transparent detection of bad disk sectors, and the relocation of data from bad to good disk sectors. The bad block reallocation feature does not exist in VxVM because the vectoring of bad blocks is now done by most hardware.

Coexistence of VxVM and LVM disks

Both LVM disks and VxVM disks can exist together on a system. The LVM disks are detected and displayed as such by VxVM. LVM disks are not selected by VxVM for initialization, addition, or replacement.

The `vxconvert` command is provided to enable LVM disks to be converted to a VxVM disk format without losing any data. It also includes support for the migration of JFS or JFS2 to VxFS.

See “[About converting LVM, JFS and JFS2 configurations](#)” on page 395.

Converting LVM, JFS and JFS2 to VxVM and VxFS

About converting LVM, JFS and JFS2 configurations

This chapter explains how to convert your LVM, JFS and JFS2 configuration to a VxVM and VxFS configuration and presents the following main topics:

- [Initializing unused LVM physical volumes to VxVM disks](#)

- [Converting LVM volume groups to VxVM disk groups](#)
The conversion process also includes the conversion of JFS and JFS2 file systems stored in LVM volume groups to VxFS.
- [Restoring the LVM volume group configuration](#)
- [Examples of using vxconvert](#)

The basic tools for conversion are the VxVM commands, `vxconvert` and `vxdiskadm`. The discussion here details how to use these tools and gives some insights into how these tools work.

The disks on your system managed by LVM can be of two types: LVM disks in volume groups, and unused disks.

The former are disks that contain logical volumes and volume groups. Unused disks contain no user data, and are not used by any volume group, but have LVM disk headers written by `cfgmgr`. Conversion is done differently for these two types of disks.

For unused LVM disks you can use `vxdiskadm`. For LVM disks in volume groups, the primary tool for conversion is the `vxconvert` command.

See the man page `vxdiskadm(1M)`.

See the *Veritas Volume Manager Administrator's Guide*.

Initializing unused LVM physical volumes to VxVM disks

LVM disks that are not part of any volume group and contain no user data are cleaned up, so that there are no LVM disk headers. Then the disks are put under VxVM control through the normal means of initializing disks.

Warning: You must be absolutely certain that the disks are not in use in any LVM configuration. If there is any user data on these disks, it will be lost during initialization.

Removing LVM disk information

To remove LVM disk header information from the disks, use the following command:

```
# chpv -C diskname
```

where *diskname* is any physical disk, such as `hdisk4`.

Initializing disks for VxVM use

To initialize the disk for VxVM use, use the `vxdiskadm` command, selecting the option:

```
1) Add or initialize one or more disks
```

Or use the command:

```
# vxdisk init disk_name
```

VxVM utilities will not tamper with disks that are recognized as owned by LVM (by virtue of the LVM VGRA disk headers). The `vxdisk init` or `vxdiskadm` commands fail if you attempt to use them on an LVM disk without first using the `chpv` command.

Converting LVM volume groups to VxVM disk groups

It is recommended that you read through this section carefully before beginning any volume group conversion.

A mounted JFS or JFS2 file system cannot be converted. Unmount such file systems before proceeding with the conversion.

The `vxconvert` process converts LVM volume groups to VxVM disk groups in the default format. You can use the `vxdiskadm` menu to specify the default format. If you do not specify a default format, `vxconvert` uses the format that is compatible with the Cross-platform Data Sharing (CDS) feature (`cdsdisk` format).

See the *Veritas Storage Foundation Cross-Platform Data Sharing Administrator's Guide*.

This section outlines the process for converting LVM volume groups to VxVM disk groups. During the conversion process, all JFS or JFS2 file systems in a specified LVM volume group are converted to VxFS.

The conversion process involves many steps. Though there are tools to help you with the conversion, some of these steps cannot be automated. You should be sure to understand how the whole conversion process works, and what you will need to do in the process before beginning a volume group conversion.

The tool used for conversion is `vxconvert`. This interactive, menu-driven program walks you through many of the steps of the process of converting volume groups for use by VxVM. Using `vxconvert` can reduce the downtime associated with converting from LVM to VxVM. Without the `vxconvert` tool, the only possible method of conversion would be to take full backups of user data, destroy the existing LVM configuration leaving only raw disks, recreate the configuration in VxVM, and then reload the user data.

The `vxconvert` process converts LVM volume groups to VxVM disk groups in place. This means, that the utility changes disks within LVM volume groups to VxVM disks by taking over the areas of the disks used for LVM configuration information, and creating the equivalent VxVM volume configuration information. User data, the portions of the disks used for databases, and so on, are not affected by the conversion. Both JFS and JFS2 data is converted during the conversion.

The act of conversion changes the names by which your system refers to the logical storage. Therefore, the conversion process is necessarily performed off-line. There can be no application access to user data in the volume groups undergoing conversion. Access to the LVM configuration itself (the metadata of LVM) must also be limited to the conversion process.

Volume group conversion limitations

There are certain LVM volume configurations that cannot be converted to VxVM. Some of the reasons a conversion could fail are:

- A volume group with insufficient space for metadata.
In the conversion of LVM to VxVM, the areas of the disks used to store LVM metadata are overwritten with VxVM metadata. If the VxVM metadata that needs to be written will not fit the space occupied by the LVM metadata, the group containing the disk cannot be converted. If you have just enough space for the conversion, you probably would want to have more space for future configuration changes.
- A volume group containing the root volume.
The current release of VxVM on AIX does not support VxVM root volumes. Because of this, `vxconvert` does not convert any volume group that contains a rootable volume. Not only is the current root volume off limits, but any volume that might be used as an alternate root volume is rejected as well.
- A volume group containing mirrors using the Mirror Write Cache feature for volume consistency recovery.
You should be aware that when converting mirrored LVM volumes to VxVM, some of these volumes will likely have the Mirror Write Cache consistency recovery method in force on the volume. The `vxconvert` utility can convert these volumes, but in some cases, it might not be able to create an equivalent level of consistency. Therefore, `vxconvert` will detect this case and warn the user that converting this volume group would lose this MWC functionality and leave the resultant VxVM mirrored disk group operating in a comparatively degraded state.
- Volume groups with any `dump` or primary `swap` volumes.
Because VxVM does not support rootability, `vxconvert` will not convert primary swap or paging space on any type of volume to VxVM.

- A volume group containing the `/usr` file system.
 For this release, a volume group containing the `/usr` file system cannot be converted because `vxconvert` needs access to files in `/usr`.
- Volume groups with any disks that have bad blocks in the bad block directory.
 Unlike LVM, VxVM does not support bad block revectoring at the physical volume level. If there appear to be any valid bad blocks in the bad block directory of any disk used in an LVM volume group, the group cannot be converted.
 The list of conversion error messages describe the actions to take in this situation.
 See [“List of conversion error messages”](#) on page 503.
- Not enough disk space on the root LVM volume group to save a copy of each physical disks VGRA area.
 For large LVM volume groups, especially those with large VGDA sizes, the required space could be greater than 64MB per physical volume. So, for a Volume Group with 128 disks, the required storage space could be greater than 8 GB.
 The default save area is `/etc/vx/reconfig.d`.
- Volume groups with mirrored volumes.
 A conversion fails if the LVM volume group being converted has mirrored volumes, but the system does not have a valid license installed that enables mirroring for VxVM.

Conversion process summary

Several steps are used to convert LVM volume groups to VxVM disk groups. Most of these steps can be done with the `vxconvert` utility. All the steps are not compulsory, and some may have to be followed only if there are problems during conversion. Some of them (e.g. backing up user data) are left to you to accomplish through your regular administrative processes.

The order of the steps in the conversion process is:

- Identify LVM volume groups for conversion.
- Analyze an LVM volume group, and then analyzing JFS or JFS2 file systems, if any, on the volume group to see if conversion is possible.
- Take action to make conversion possible if analysis fails.
- Back up your LVM configuration and user data.
- Plan for new VxVM logical volume names.
- Stop application access to volumes in the volume group to be converted.

- Convert the JFS or JFS2 file systems, if any, on a specified volume group, and then converting the volume group.
- Take action if conversion fails.
- Implement changes for new VxVM logical volume names.
- Restart applications on the new VxVM volumes.
- Tailor your VxVM configuration.

These steps are described in detail in later sections of this section, including examples of how to use `vxconvert`.

See [“Examples of using vxconvert”](#) on page 409.

You can also restore back to your original LVM configuration.

See [“Restoring the LVM volume group configuration”](#) on page 409.

Conversion of JFS and JFS2 file systems to VxFS

The `vxconvert` utility converts JFS and JFS2 file systems to VxFS file systems with a Version 7 disk layout.

Conversion from a JFS or JFS2 file system to a VxFS file system requires that there is sufficient free space within the file system or immediately after the end of the file system to convert the existing metadata. The space must be available on the same device or volume on which the file system resides. The amount of free space that is required is approximately 12-15% of the total size of the file system size, but the exact amount depends on the number and sizes of files and directories, and on the number of allocated inodes. The conversion process takes up to 3 times longer than running a file system check (`fsck`) on the file system.

After conversion, you can use utilities such as `fsadm` and `vxresize` to reorganize the file system.

The ability to shrink a file system that has been converted to VxFS depends on the amount and location of the remaining free space in the file system. If an attempt to shrink a converted file system fails, specify a smaller shrink size.

JFS or JFS2 log devices and JFS2 snapshot devices are not touched by the conversion process. After the file systems are converted, you can recover the space used by these devices for other purposes.

Conversion limitations

The following conversion limitations must be considered:

- You cannot use the `vxconvert` utility to reverse the conversion of a JFS or JFS2 file system to VxFS. Instead, you must recreate the original file system and restore the data from a backup.
- Compressed JFS file systems cannot be converted. You must first decompress a compressed JFS file system before starting the conversion process.
- A JFS file system with a fragment size of 512 bytes cannot be converted.
- The quota files in JFS or JFS2 file systems are not converted to the VxFS quota file format.
- The extended attributes of a JFS file system are not converted to VxFS extended attributes.
- A JFS2 file-system with a block size of 512 bytes cannot be converted.
- A JFS2 file system with inode numbers larger than 2^{32} cannot be converted.
- JFS2 v1 file systems with extended attributes can be converted, but these attributes are not preserved.
- JFS2 v2 file systems with named attributes can be converted, but ACLs and DMAPI attributes are not preserved.
- JFS2 file systems with snapshots may be converted, but the snapshots are not preserved.

Conversion steps explained

Perform the following steps in this order for the conversion:

- [Identify LVM disks and volume groups for conversion](#)
- [Analyze an LVM volume group to see if conversion is possible](#)
- [Take action to make conversion possible if analysis fails](#)
- [Back up your LVM configuration and user data](#)
- [Plan for new VxVM logical volume names](#)
- [Stop application access to volumes in the volume group to be converted](#)
- [Conversion and reboot](#)
- [Convert a volume group](#)
- [Take action if conversion fails](#)
- [Implement changes for new VxVM logical volume names](#)
- [Restart applications on the new VxVM volumes](#)
- [Tailor your VxVM configuration](#)

Identify LVM disks and volume groups for conversion

The obvious first step in the conversion process is to identify what you want to convert. The native LVM administrative utilities like `lsvg` and `SMIT` can help you identify candidate LVM volume groups as well as the disks that comprise them.

You can also use the `vxconvert` and `vxdisk` commands to examine groups and their member disks.

The information presented through the `vxconvert` and `vxdisk` utilities and their interpretation is shown in several examples.

See [“Examples of using vxconvert”](#) on page 409.

You can also list the LVM disks with the following VxVM command:

```
# vxdisk list
```

Analyze an LVM volume group to see if conversion is possible

After you have selected a volume group for conversion, you need to analyze it to determine if conversion for VxVM use is possible.

Use the `analyze` option of `vxconvert` to check for problems that would prevent the conversion from completing successfully. This option checks for several conditions.

See [“Volume group conversion limitations”](#) on page 398.

The analysis calculates the space required to add the volume group disks to a VxVM disk group, and to replace any existing disks and volumes with VxVM volumes, plexes, and subdisks. If you do not have the required space to convert the disks, the conversion fails. The analysis also calculates the space required to convert volumes containing JFS or JFS2 file systems to VxFS. If there is insufficient space in any of these volumes, the conversion is aborted.

Before you analyze an LVM volume group, the file system must be unmounted. If you try to analyze a live system, the analysis fails and you receive an error message.

To analyze LVM volume groups, choose option 1 of the `vxconvert` utility.

Note: The analysis option is presented as a separate menu item in `vxconvert`, but there is an implicit analysis with any conversion. If you simply select the “Convert LVM and JFS to VxVM and VxFS” menu option, `vxconvert` will go through analysis on any group you specify. When you are using the `convert` option directly, you are given a chance to abort the conversion after analysis, and before any changes are committed to disk.

See [“Converting LVM volume groups to VxVM disk groups”](#) on page 397.

The analysis option is useful when you have a large number of groups/disks for conversion and some amount of planning is needed before the actual conversion. Installations with many users or critical applications can use the analyze option on a running system. Then conversion downtime can be better planned and managed. Smaller configurations may be better served by using the convert option directly while in a downtime period.

Sample examples of the analyze option are shown.

See [“Examples of using vxconvert”](#) on page 409.

Take action to make conversion possible if analysis fails

Analysis may fail for several reasons.

See [“Volume group conversion limitations”](#) on page 398.

Messages from `vxconvert` will explain the type of failure and any actions that can be taken before retrying the analysis.

Details of specific error messages and actions are provided.

See [“List of conversion error messages”](#) on page 503.

Back up your LVM configuration and user data

After analysis you know which volume group or groups you want to convert to VxVM disk groups. Up to this point, you have not altered your LVM configuration.

By taking the next step (completing the conversion to VxVM), you are significantly changing access to your storage.

Although the conversion process does not move, or in any other way affect user data, you are strongly encouraged to back up all data on the affected disks.

During a conversion, any spurious reboots, power outages, hardware errors or operating system bugs can have unpredictable and undesirable consequences. You are advised to be on guard against disaster with a set of verified backups.

The `vxconvert` utility itself also saves a snapshot of the LVM metadata in the process of conversion for each disk. It can only be used via the `vxconvert` program. With certain limitations, you can reinstate the LVM volumes after they have been converted to VxVM using this data.

See [“Displaying the vxconvert main menu”](#) on page 409.

Even though `vxconvert` provides this level of backup of the LVM configuration, you are advised to back up your data before running `vxconvert`.

To back up user data, use your regular backup processes.

Warning: Before you do the backup, you should be sure that all applications and configuration files refer properly to the new VxVM logical volumes.

See [“Implement changes for new VxVM logical volume names”](#) on page 408.

Backup processes and systems themselves may have dependencies on the volume names currently in use on your system. The conversion to VxVM changes those names. You are advised to understand the implications name changes have for restoring from the backups you are about to make.

To back up data, you can use the backup utility that you normally use to back up data on your logical volumes. For example, to back up logical volumes that contain file systems, the `backup(1M)` command can be used to back up the data to tape.

For example, to backup the data mounted on `/foodir` to device `/dev/rmt#`, use the following command:

```
# backup -0 -u -f /dev/rmt# /foodir
```

To back up application information, if a logical volume you are converting does not contain a file system, and is being used directly by an application (such as a database application), use the backup facilities provided by the application. If no such facility exists, consider using the `dd` command.

Plan for new VxVM logical volume names

When you change from LVM volumes to VxVM volumes, the device names by which your system accesses data are changed. LVM creates device nodes for its logical volumes in `/dev` under directories named for the volume group. VxVM creates its device nodes in `/dev/vx/dsk` and `/dev/vx/rdsk`. When conversion is complete, the old LVM device nodes are gone from the system, and the system will access data on the device nodes in `/dev/vx`.

This change in names can present problems. Any application that refers to specific device node names will be at risk when these names change. Similarly, any files that record specific device node names for use by applications can be problematic.

The most obvious area where this problem arises is in the `/etc/filesystems` file. To handle this problem, `vxconvert` rewrites `/etc/filesystems` with the new VxVM names when conversion is complete so that `fsck`, `mount`, and related utilities will behave as they did prior to the conversion.

There are potentially many other applications, though, that may be put at risk by the name changes in conversion. `vxconvert` cannot help with these. The system administrator must examine the mechanisms used in each of the following areas to see if they reference LVM device names:

- Databases run on raw logical devices may record the name of that device node.
- Backup systems may do device level backups based on device node names recorded in private files. Also labelling of the backups may record device names.
- Scripts run by cron(1M).
- Other administrative scripts.

To work around the issue of the name changes in conversion, use the `vxconvert` mapping file. `vxconvert` records a mapping between the names of the LVM device nodes and VxVM device nodes. This data can be used to create symbolic links from the old LVM volume to the new VxVM device names. The mapping is recorded in the following file:

```
/etc/vx/reconfig.d/vgrecords/vol_grp_name/vol_grp_name.trans
```

This file provides information on how to proceed further to link the old LVM volume names to the new VxVM device names.

Warning: This method of resolving the naming problem has risks. The symbolic links can become stale. For example, if a database refers to `/dev/vx/rdisk/vol1` through a symbolic link `/dev/rvol1` (“the old LVM name”), and if the underlying VxVM volume configuration is changed in any way, the database could refer to a missing or different volume.

Note: You may want to use this symbolic link approach to ease the transition to VxVM. You can set up the symbolic links after the successful conversion to VxVM. Then, you can do the investigation on a case by case basis for each volume. When you are satisfied that there are no problems introduced by the name change, the symbolic link to that volume can be removed. You must be careful to maintain a static VxVM volume configuration during this transition period.

Over time, the ultimate goal should be that the underlying VxVM naming is used by all applications, and that there are no indirect references to those volumes.

Stop application access to volumes in the volume group to be converted

No applications can be active on the LVM volume group undergoing conversion. Before attempting to convert any volume group, you must ensure that applications using that group are down. This involves stopping databases, unmounting file systems, and so on.

Note: You need to check and update the `/etc/filesystems` file for valid and supported options for the VxFS file systems before mounting.

During the conversion, `vxconvert` tries to unmount mounted file systems.

See “[Conversion and reboot](#)” on page 406.

However, `vxconvert` makes no attempt to close down running applications on those file systems, also, it does not attempt to deal with applications (e.g., databases) running on raw LVM volumes.

Note: It is strongly recommended that you do not rely on `vxconvert`'s mechanisms for unmounting file systems. Conversion will be simpler if you close applications, and unmount file systems before running `vxconvert`.

To unmount a file system, use the following command:

```
# umount file_system_device
```

For example:

```
# umount /dev/lv01
```

Having unmounted a file system, use the `fsck` command to check its integrity:

```
# fsck -y file_system_device
```

For example:

```
# fsck -y /dev/lv01
```

Conversion and reboot

During conversion, after the analysis phase is complete, the disks to be converted are deemed to be conversion ready. The `vxconvert` program asks if you are ready to commit to the conversion changes. If you choose to complete the conversion, the system will try to unmount all of the associated mounted file systems, stop and export the volume group, and then install the VxVM configuration.

If `vxconvert` is unable to stop and export volume groups or unmount file systems, the conversion cannot be completed without rebooting the system. You will have the option of aborting the conversion or completing the conversion by rebooting the system. If you choose to reboot, `vxconvert` will trigger the completion of the conversion automatically, during reboot, when it can be guaranteed that no processes have access to the volumes that are being converted.

If you choose to abort rather than reboot to complete the conversion, `vxconvert` will return to the main menu.

Note: The LVM logical volumes to be converted must all be available to the `vxconvert` process. You should not deactivate the volume group or any logical volumes before running `vxconvert`.

To activate a volume group when you are not certain if the LVM volumes or the corresponding volume groups are active, you can activate them with the following command:

```
# varyonvg vol_grp_name
```

Convert a volume group

To do the actual conversion of LVM volume groups to VxVM disk groups, choose option 2 of the `vxconvert` utility.

`vxconvert` will prompt for a name for the VxVM disk group that will be created to replace the LVM volume group you are converting. This is the only object naming that is done through `vxconvert`.

VxVM volume names may need modified.

See [“Tailor your VxVM configuration”](#) on page 408.

The volume groups selected for conversion are analyzed to ensure that conversion is possible.

See [“Analyze an LVM volume group to see if conversion is possible”](#) on page 402.

After a successful analysis phase, `vxconvert` will ask you to commit to the change or abort the conversion. When you select to commit to conversion, the new VxVM metadata is written.

Note: It is good practice to convert one volume group at a time to avoid errors during conversion.

Examples with details of the conversion process are available.

See [“Examples of using vxconvert”](#) on page 409.

Take action if conversion fails

Conversion can fail for many reasons.

See [“Volume group conversion limitations”](#) on page 398.

Messages from `vxconvert` will explain the type of failure, and any actions you can take before retrying the conversion.

See “[List of conversion error messages](#)” on page 503.

Implement changes for new VxVM logical volume names

You must be sure that all applications and configuration files refer properly to the new VxVM logical volumes.

See “[Plan for new VxVM logical volume names](#)” on page 404.

Restart applications on the new VxVM volumes

After the conversion to VxVM is complete, file systems can be mounted on the new devices and applications can be restarted.

For the file systems that you unmounted before running `vxconvert`, remount them using the new volume names. `vxconvert` will have updated `/etc/filesystems` with the new names.

After conversion, remove any VxVM log volumes that have been converted from corresponding JFS or JFS2 log volumes.

Tailor your VxVM configuration

`vxconvert` provides a default name for naming the newly formed VxVM disk group during conversion only as an option. However, you will be given the choice of choosing your own VxVM disk group name. By default, `vxconvert` renames the LVM volume group by replacing the prefix `vg` in the volume group name with the prefix `dg`. For example, `vg08` would become `dg08`. If there is no `vg` in the LVM volume group name, `vxconvert` simply uses the same volume group name for its disk group.

The disks in the new VxVM disk group are given VxVM disk media names based on this disk group name.

See the man page `vxintro(1M)`.

If your new VxVM disk group is `dg08`, it will have VxVM disks with names like `dg0801`, `dg0802`, and so on. The VxVM plexes within the logical volumes will be `dg0801-01`, `dg0801-02`, and so on.

If you do not like the default object names generated by the conversion, use the standard VxVM utilities to rename these objects. See the `rename` option in the `vxedit(1M)` man page for more details on renaming the disk groups.

Note: You must only rename objects in the VxVM configuration after you are fully satisfied with that configuration.

If you have chosen to set up symbolic links to the VxVM volumes, avoid renaming VxVM objects.

See “[Plan for new VxVM logical volume names](#)” on page 404.

These symbolic links are made invalid if the underlying VxVM device node name changes.

Restoring the LVM volume group configuration

If you need to restore the original LVM configuration, you must restore the user data in addition to restoring the old LVM metadata and associated configuration files.

Note: The snapshot of LVM internal data is kept on the `root` file system. You must have backed up data located on all the volume groups’ logical volumes before conversion to VxVM.

Restoration of LVM volume groups is a two-step process consisting of a restoration of LVM internal data (metadata and configuration files), and restoration of user or application data.

Examples of using vxconvert

The following sections contain examples of using the `vxconvert` utility.

Displaying the vxconvert main menu

To display the `vxconvert` menu, use the following command:

```
# vxconvert
...
Press return to continue

VERITAS Storage Foundation Operations
Menu: Volume Manager and File System Conversion

1      Analyze LVM Volume Groups and JFS/JFS2 File Systems
        for Conversion
2      Convert LVM and JFS/JFS2 to VxVM and VxFS
```

```
3      Set path for saving VGRA records and JFS/JFS2 meta
      data
list   List disk information
listvg List LVM Volume Group information

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

Select an operation to perform:

Listing disk information

The `list` option of `vxconvert` displays information about the disks on a system.

...

Select an operation to perform: # **list**

List disk information

Menu: Volume Manager/LVM_Conversion/ListDisk

Use this menu option to display a list of disks. You can also choose to list detailed information about the disk at a specific disk device address.

Enter disk device or "all" [<address>,all,q,?] (default: all) **all**

DEVICE	DISK	GROUP	STATUS
EXT_DISKS_0	-	-	LVM
EXT_DISKS_1	-	-	online invalid
EXT_DISKS_2	-	-	online invalid
EXT_DISKS_3	-	-	online invalid
EXT_DISKS_4	-	-	online invalid
EXT_DISKS_5	-	-	online invalid
EXT_DISKS_6	-	-	LVM
EXT_DISKS_7	-	-	online invalid
EXT_DISKS_8	-	-	LVM
EXT_DISKS_9	disk01	rootdg	online invalid

Device to list in detail [<address>,none,q,?] (default: none) **none**

Listing LVM volume group information

To list LVM volume group information, use the `listvg` option.

Note: The volume groups you want to convert must not be a root volume group or have bootable volumes in the group.

Analyzing LVM volume groups, JFS and JFS2 for conversion

To analyze one or more LVM volume groups and any JFS or JFS2 file systems, select option 1.

Example of failed analysis for LVM volumes

The following example shows a failed analysis for LVM volumes.

```
Second Stage Conversion Analysis of vgbig
```

```
There is not enough free space on the /etc/vx device
to complete the conversion of the LVM Volume Group (vgbig).
You will need to have at least 17530 blocks free.
```

Converting LVM volume groups, and JFS or JFS2 file systems

To convert LVM volume groups to VxVM disk groups, and JFS or JFS2 file systems to VxFS, select option 2.

Example of a failed JFS conversion

The following example shows a failed JFS conversion.

```
Found insufficient space to convert the JFS (/dev/lv01) to VxFS.
At least 2972 kilobytes space is required. To allow conversion,
please go back and increase the filesystem size.
```

```
Analysis of (/dev/lv01) from JFS to VXFS has failed due to the
error:
```

```
vxfs vxfsconvert: Total of 2972K bytes required to complete
the conversion
```

```
Please check the error and do accordingly.
```

Example of a failed LVM conversion

The following example shows a failed LVM conversion.

```
Second Stage Conversion Analysis of vgbig
```

```
There is not enough free space on the /etc/vx device
```

to complete the conversion of the LVM Volume Group (vgbig).
 You will need to have at least 17530 blocks free.

Sample output before and after conversion

The following example show output before and after conversion.

Before conversion

The following example shows output before conversion.

After conversion

The following example shows output after conversion.

```
# vxconvert
...
Select an operation to perform: listvg
...
Enter Volume Group (i.e.- vg04) or "all"
[<address>,all,q,?] (default: all)
```

```
LVM VOLUME GROUP INFORMATION
Name      Type      Physical Volumes
rootvg    ROOT      hdisk0
...
Select an operation to perform: q
```

```
# vxprint
Disk group: rootdg
```

TY	NAME	ASSOC	KSTATE	LENGTH	PLOFFS	STATE	TUTIL0	PUTIL0
dg	rootdg	rootdg	-	-	-	-	-	-
dm	disk01	EXT_DISKS_9	-	35589724	-	-	-	-

Disk group: dgbig

TY	NAME	ASSOC	KSTATE	LENGTH	PLOFFS	STATE	TUTIL0	PUTIL0
dg	dgbig	dgbig	-	-	-	-	-	-
dm	dgbig01	EXT_DISKS_6	-	17765832	-	-	-	-

v	loglv00	gen	ENABLED	131072	-	ACTIVE	-	-
pl	loglv00-01	loglv00	ENABLED	131072	-	ACTIVE	-	-
sd	dgbig01-01	loglv00-01	ENABLED	131072	0	-	-	-
v	lv01	fsgen	ENABLED	655360	-	ACTIVE	-	-
pl	lv01-01	lv01	ENABLED	655360	-	ACTIVE	-	-

sd	dgbig01-03	lv01-01	ENABLED	655360	0	-	-	-
v	lv02	fsgen	ENABLED	65536	-	ACTIVE	-	-
pl	lv02-01	lv02	ENABLED	65536	-	ACTIVE	-	-
sd	dgbig01-02	lv02-01	ENABLED	65536	0	-	-	-

The disk group `dgbig` contains the VxVM disk `dgbig01` and the volume `loglv00`. The VxVM disk `dgbig01` is associated with disk device `EXT_DISKS_6` and is 17765832 blocks in length. The volume `loglv00` is of type `gen`, is enabled in the VxVM kernel driver, is of length 131072, and is in the `ACTIVE` state. This means that the volume is started, and the plex is enabled. Operations to the volume such as recovery and data access will be governed by the usage type `gen`.

The plex `loglv00-01` is associated with volume `loglv00`, and maps the entire address range of the volume. Associated with the plex is one subdisk, `dgbig01-01` which maps the plex address range from 0 to the entire length of the plex, that is, 131072 blocks. As implied by the first part of its name, the subdisk `dgbig01-01` uses an extent from the VxVM disk `dgbig01`.

General information regarding conversion speed

Factors affecting conversion speed include:

- Size of volume groups. The larger the volume groups, the larger the VGRA area on each disk. A copy must be made of the VGRA area of each physical disk. Some areas are greater than 64MB; therefore a 50-disk volume requires 64MB reads and writes (that is, 100 large I/O requests) to complete. Some volume groups have 128 disks.
- Individual size of a logical volume in a volume group, and the complexity of the logical volume layout. For example, for a system with 50 9GB drives, a simple 50GB logical volume can be created using 6 disks. But a 50GB striped logical volume that takes the first 1GB of all 50 disks can also be created. The first and simple logical volume takes less time to convert than the striped volume since only 5 disks need to be checked for metadata. However, for the striped volume, 50 disks need to be checked and 50 VGRAs to be copied. In addition, the complexity of reproducing the VxVM commands to set up the striped volumes requires more VxVM commands to be generated to represent more smaller subdisks representing the same amount of space. Another factor in converting stripes is that stripes create more work for the converter. In some cases, stripes require 1GB volume, although only the metadata is being changed. In other cases, where there are more physical disks in one volume than another, there is more metadata to deal with. The converter has to read every physical extent map to ensure there are no holes in the volume; if holes are found, the converter maps around them.

- Number of volumes. While it takes longer to convert one 64GB volume than one 2GB volume, it also takes longer to convert 64 1GB volumes than one 64GB volume, providing that the volumes are of similar type.
- Mirrored volumes. Mirrored volumes typically do not take more time to convert than simple volumes. Volumes that are mirrored and striped at the same time would take longer.

Currently, after conversion, mirrored volumes are not automatically synchronized because a large mirror could take hours to complete.

For example, in tests, a 150 GB volume group consisting of 20 simple logical volumes takes approximately 35-40 minutes to convert. In contrast, the same volume group (150GB) consisting of mirrored volumes that need to be synchronized can take 30-40 hours to convert.

Note: If you convert mirrored volumes, you must synchronize them in a separate step.

Estimating system down time

When you want to convert your LVM configurations to VxVM, a common question is, “How long will it take?” Symantec has developed a series of test cases to evaluate various configurations and calculate conversion times. Based on this data, Symantec can reasonably estimate your down time.

About test cases

This section describes the test case configuration and the factors that impact conversion time.

Test case configuration

The test cases use the following configuration:

Storage Foundation (SF) release version	SF 5.0-MP3
Hardware configuration	Power 5 H/W, Model-51A
Operating system level	AIX 5.3 TL07 SP2
Storage array	EMC Clariion with EMC Powerpath enabled

Factors that impact conversion time

The test cases were formulated based on several theoretical factors that impact the conversion time, including:

- The average size of a file in the file system
- The size of the file system
- The number of files part of the file system
- The number of physical volumes in the volume group
- The number of logical volumes in the volume group

Test case: file number and size, file system size

This test investigates the following factors:

- The number of files in the file system
- The average size of a file in the file system
- The size of the file system

Table 27-1 Values used in the test case

File	Ave. file size (GB)	File system size (GB)	LVM volume group size (GB)	Conversion time (min:sec)
4	5	50 (jfs)	50	1:28
10	3	70 (jfs)	180	3:12
800	.50	450 (jfs2)	450	3:36
9	42	400 (jfs2)	400	3:49

Inference

Based on the test results, you can infer the following:

- Decreasing the number of files decreases the conversion time.
- Increasing the free space in the file system decreases the conversion time.

Test case: file number, file system size

This test case investigates the following factors:

- The number of files in the file system varies, while the file size is kept constant at 0.04 MB.

- The size of the file system.

Table 27-2 Values used in the test case

Files	File system size (GB)	Disks	Logical volumes	Volume group size (GB)	Conversion time (min:sec)
150	5	3	3	6	2:44.281
25000	5	3	3	6	3:39.377
150	15	8	3	16	4:22.730
25000	15	8	3	16	5:38.280
100000	5	4	3	6	8:22.436
100000	15	8	3	16	9:01.918
150	50	29	2	59	24:18.791
25000	50	29	2	59	26:24.359
100000	50	26	2	51	27:05.961

Inference

Based on the test results, you can infer that increasing the number of files in a file system size range increases the conversion time.

Note: In this test, the increase in conversion time is marginal compared to the increase in the number of files. This can be attributed to the fairly small file system size. During one of the customer data center migrations from LVM to VxVM, there was a significant increase in the conversion time when a large file system (about 2 TB) had a very large number of files (nearly 2 million files).

Test case: average file size

This test investigates how the average size of a file in the file system affects the conversion time. The amount of data, that is, the number of files multiplied by the average size of each file, remains constant. As a result, as the file size decreases, the number of files increases.

The test was performed while keeping the following parameters constant:

Number of volumes

1

Average volume size	350 GB
File system size	350 GB (jfs)
Number of disks	8
LVM volume group size	400 GB
Number of volume groups	1

Table 27-3 Values used in the test case

Files	Ave. file size (GB)	Conversion time (min:sec)
24	12.5	4:21.412
100	3	4:25.849
48	6.25	4:26.339
300	1	4:41.521
12	25	4:46.357
600	.50	5:04.095
6	50	5:15.490
1200	.25	5:25.575

Inference

Based on the test results, you can infer the following:

- Decreasing the file size reduces the conversion time
- Increasing the number of files toward the end of the test increases conversion time

Test case: number of logical volumes

This test investigates how the number of logical volumes (LVs) in the volume group affects the conversion time.

You should compare these test results with the average file size test case.

See “[Test case: average file size](#)” on page 416.

This test was performed while keeping the following parameters constant:

Number of volumes	24 + 1
-------------------	--------

Average volume size	1 300 GB LV + 24 2 GB LVs
File system size	350 GB (jfs)
Number of disks	7
LVM volume group size	400 GB
Number of volume groups	1

Table 27-4 Values used in the test case

Files	Ave. file size (GB)	Conversion time (min:sec)
300	1	5:11.013
600	.50	5:26.123
1200	.25	5:42.171
24	12.5	6:22.357
48	6.25	6:26.523
100	3	6:41.312
12	25	6:42.512
6	50	7:05.872

Inference

Based on the test results, you can infer that increasing the number of logical volumes increases the conversion time; however the increase is marginal.

Test case: number of physical volumes

This test investigates how the number of physical volumes in the volume group affects conversion time.

The test was performed while keeping the following parameters constant:

Number of disks	52
Number of logical volumes	1
Size of each file	1 MB
Volume group size	100 GB

Table 27-5 Values used in the test case

File system size (GB)	Files	Conversion time (min:sec)
5	150	83:57.106
5	25000	84:42.825
5	100000	80:05.713
15	150	81:03.553
15	25000	83:37.725
15	100000	84:51.357
50	150	82:04.489
50	25000	85:01.479
50	100000	85:23.097
100	150	79:24.965
100	25000	96:13.457
100	100000	93:59.675

Inference

Compared with the test case that focuses on the file number and file system size, when the number of disks is kept constant across the test case, conversion time "leap frogs." This result implies that increasing the number of disks in a volume group plays a significant role in increasing the conversion time.

See "[Test case: file number, file system size](#)" on page 415.

Command differences

Command differences between LVM and VxVM

This chapter describes the differences between LVM and VxVM commands, and tasks. It includes a task comparison chart which lists some of the tasks that are performed using LVM with a near equivalent task performed using VxVM. It also provides a list of VxVM tasks which are not available with LVM, and the LVM features currently not supported in VxVM.

For more information on LVM commands, refer to *AIX Logical Volume Manager, from A to Z: Introduction and Concepts*. For information on VxVM commands, refer to the *Veritas Volume Manager Administrator's Guide*.

LVM and VxVM command equivalents

[Table 27-6](#) lists the LVM commands and a near equivalent command to use in VxVM.

For more information, refer to the task comparison chart.

See “[Comparison of LVM and VxVM tasks](#)” on page 424.

For information on VxVM commands, refer to the *Veritas Volume Manager Administrator's Guide*.

Table 27-6 Command comparison

LVM	Description/action	VxVM	Description/action
chlv	Changes the characteristics of logical volumes.	vxedit vxvol set	Creates, removes, and modifies Volume Manager records.
	There is no single equivalent LVM command.	vxresize	Resizes a file system and its underlying volume at the same time.
mklv	Creates a logical volume.	vxassist	Creates volumes with the make parameter. Example: <pre>vxassist make \ vol_name 100M \ layout=stripe</pre>

Table 27-6 Command comparison (*continued*)

LVM	Description/action	VxVM	Description/action
<code>extendlv</code>	Increases disk space allocated to a logical volume.	<code>vxassist</code>	Increases a volume in size with the <code>growto</code> parameter or the <code>growby</code> parameter. Example: <code>vxassist growto \ vol_name 200M vxassist growby \ vol_name 100M</code> <code>vxassist</code> creates and modifies volumes.
<code>syncvg -l</code>	Synchronizes mirrors that are stale in one or more logical volumes.	<code>vxrecover vxvol start</code>	The <code>vxrecover</code> command performs resynchronize operations for the volumes, or for volumes residing on the named disks (<code>medianame</code> or the VxVM name for the disk). Example: <code>vxrecover \ vol_name media_name</code>
<code>lspv</code>	Displays information about physical volumes in a volume group.	<code>vxdisk list</code>	Lists information about VxVM disks. Example: <code>vxdisk list \ disk_name</code>

Table 27-6 Command comparison (*continued*)

LVM	Description/action	VxVM	Description/action
chpv	Sets physical volume characteristics to allow/deny allocation of additional physical extents from this disk.	vxdisk vxdisk set vxedit	The <code>vxdisk</code> utility performs basic administrative operations on VxVM 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.
chpv -C	Removes the LVM header information and releases the disk from LVM control.	vxdiskunsetup	Removes the VxVM header information and releases the disk from VxVM control.
mkvg	Creates a volume group.	vxdiskadd vxdg init	Creates a new disk group and/or adds disks to a disk group.
lsvg	Displays information on all volume groups.	vxdg list vxprint	<code>vxdg list</code> displays the contents of a disk group. <code>vxprint</code> displays information about all objects or a subset of objects.
chvg	Activates or deactivates one or more volume groups.	vxdg -g \ <i>diskgroup</i> set \ activation= <i>mode</i>	Activates a shared disk group.
extendvg	Extends a volume group by adding one or more disks to it.	vxdiskadd vxdiskadm	<code>vxdiskadd</code> adds a disk to the disk group. For <code>vxdiskadm</code> , use the option <code>Add</code> or initialize one or more disks to add disks to the disk group.

Table 27-6 Command comparison (*continued*)

LVM	Description/action	VxVM	Description/action
reducevg	Reduces a volume group by removing one or more disks from it.	vxvg rmdisk vxdiskadm	vxvg removes disks from a disk group. For vxdiskadm, use the option Remove a disk to remove disks.
lsvg	Scans all disks and looks for logical volume groups.	vxinfo vxprint vxdiskadm	vxinfo displays information about volumes. vxprint displays complete or partial information from records in VxVM disk group configurations. For vxdiskadm, use the option list to display disk information.
syncvg	Synchronizes mirrors that are stale in one or more logical volumes.	vxrecover	Starts resynchronization and recovery of volumes.
reducevg	Removes the definition of a volume group from the system.	vxvg deport vxdiskadm	Deports a disk group from the system. For vxdiskadm, use the option Remove access to (deport) a disk group menu to remove a disk group.
exportvg	Removes a volume group from the system.	vxvg deport vxdiskadm	Deports a disk group from the system. For vxdiskadm, use the option Remove access to (deport) a disk group menu to remove a disk group.

Table 27-6 Command comparison (*continued*)

LVM	Description/action	VxVM	Description/action
<code>importvg</code>	Adds a volume group to the system by scanning the physical volumes which have been exported using <code>vgexport</code> .	<code>vxdg import</code> <code>vxdiskadm</code>	Imports a disk group. For <code>vxdiskadm</code> , use the option <code>Enable access to (import)</code> a disk group to import a disk group.
	No LVM command	<code>vxplex</code>	Operates on plex objects.
<code>chlv</code> , <code>extendlv</code> , <code>mklv</code> , <code>rmlv</code>	Performs operations on logical volumes.	<code>vxvol</code>	Operates on volume objects.
	No LVM command	<code>vxsd</code>	Operates on subdisk objects.
	No LVM command	<code>vxmend</code>	Fixes simple misconfigurations.

Comparison of LVM and VxVM tasks

This section contains a list of tasks which you can perform using LVM, and near equivalent tasks which you can perform using Veritas Volume Manager. You can perform the LVM tasks by using SMIT or the command line interface. Similarly, you can choose to perform VxVM tasks by using the Storage Foundation Manager (SFM) or the command line interface. This section focuses on the command line interface.

The following features in VxVM require an additional license:

- Mirroring
- Mirroring and Striping
- Dynamic Multipathing of Active/Active Devices
- Hot-relocation
- Online Migration
- RAID-5

For more information on LVM commands, refer to the following document:

AIX Logical Volume Manager from A-Z: Troubleshooting and Commands.

For information on VxVM commands, refer to the *Veritas Volume Manager Administrator's Guide*.

Note: VxVM is not rootable for this release, so you cannot perform these tasks on the root disk or root volume.

Table 27-7 shows the command for bringing a disk under VxVM control.

There is no equivalent LVM task.

Table 27-7 Bringing a disk under VxVM control

Task type	Description	Example
VxVM	Bring a disk under Volume Manager control.	<code>vxdiskadd device_name</code> Use option 1 in the <code>vxdiskadm</code> menu to add a disk and initialize it.

Table 27-8 shows the commands for creating volume and disk groups.

Table 27-8 Creating groups

Task type	Description	Example
LVM	Create a volume group	<code>mkvg [-y vol_grp] PV_name</code>
VxVM	Create a disk group.	<code>vxvg init disk_group \ disk_name....</code> Use option 1 in the <code>vxdiskadm</code> menu to perform this task.

Table 27-9 shows the commands for adding disks.

Table 27-9 Adding disks

Task type	Description	Example
LVM	Add a new disk to the existing volume group.	<code>extendvg vol_grp PVname....</code>
VxVM	Add a disk to an existing disk group.	<code>vxvg -g disk_group adddisk \ [disk=]device....</code>

Table 27-10 shows the commands for extending or reducing volumes.

Table 27-10 Extending or reducing volumes

Task type	Description	Example
LVM	Extend a logical volume or increase space allocated to a logical volume.	<code>extendlv <i>lvol_name</i> <i>NumberOfLPs</i></code>
VxVM	Increase the volume by or to a given length.	<pre> vxresize -g <i>disk_group</i> \ -F vxfs <i>vol_name</i> <i>length</i> vxassist -g <i>disk_group</i> \ -b growto <i>vol_name</i> <i>new_length</i> vxassist -g <i>disk_group</i> -b growby <i>vol_name</i> \ <i>length_change</i> </pre> <p>Grow the file system after growing the volumes.</p>
VxVM	Reduce a volume by or to a given length.	<pre> vxresize -g <i>disk_group</i> \ -F vxfs <i>vol_name</i> <i>to_length</i> vxassist -g <i>disk_group</i> \ -b shrinkby <i>vol_name</i> <i>length</i> vxassist -g <i>disk_group</i> \ -b shrinkto <i>vol_name</i> <i>newlength</i> </pre> <p>Shrink the file system before reducing the volume.</p>

Table 27-11 shows the commands for importing and deporting objects.

Table 27-11 Importing and deporting objects

Task type	Description	Example
LVM	Import and activate a volume group.	<pre> importvg [-y <i>vol_grp</i>] \ <i>disk_name</i> varyonvg <i>vol_grp</i> </pre>

Table 27-11 Importing and deporting objects (*continued*)

Task type	Description	Example
VxVM	Import a disk group to make the specified disk group accessible on the local machine.	<code>vxdg -tfc [-n newname] \ import disk_group</code> Use option 7 in the <code>vxdiskadm</code> menu to perform this task.
LVM	Export and deactivate an LVM volume group, and its associated logical volumes.	<code>varyoffvg vol_group exportvg vol_group</code>
VxVM	Deport a disk group to disable access to the specified disk group. A disk group cannot be deported if any volumes in the disk group are currently open.	<code>vxdg deport disk_group</code> Option 8 in the <code>vxdiskadm</code> menu performs this task.

[Table 27-12](#) shows the commands for removing groups.

Table 27-12 Removing groups

Task type	Description	Example
LVM	Remove a volume group. This destroys a volume group by removing its last disk and removing it from <code>/etc/filesystems</code> .	1. Delete all logical volumes to that volume group using the <code>rmlv</code> command. 2. Reduce the volume group using the following command: <code>reducevg [-d] [-f] VGname \ PVname...</code> The volume group is deleted when the last PV (disk) is removed from the volume group.
VxVM	Destroy a disk group.	<code>vxdg destroy disk_group</code>

[Table 27-13](#) shows the commands for removing disks.

Table 27-13 Removing disks

Task type	Description	Example
LVM	Reduce a volume group by reducing the number of disks in a volume group.	<code>reducevg [-d] [-f] vol_grp \ PVname...</code>
VxVM	Remove a disk from disk group.	<code>vxvg -g disk_group -k rmdisk \ disk_name...</code>

[Table 27-14](#) shows the commands for creating volumes.

Table 27-14 Creating volumes

Task type	Description	Example
LVM	Create a logical volume in LVM volume group.	<code>mklv -y vol_name -t type \ vol_grp #</code>
VxVM	Create a concatenated volume	<code>vxassist -g disk_group make \ vol_name length</code>
VxVM	Create a striped mirror volume.	<code>vxassist -g disk_group make \ vol_name length \ layout=mirror, stripe</code>
VxVM	Create a RAID-5 volume.	<code>vxassist -g disk_group make \ vol_name length \ layout=raid5</code>

[Table 27-15](#) shows the commands for displaying volume and disk group information.

Table 27-15 Displaying volume and disk group information

Task type	Description	Example
LVM	Display information about logical volumes.	<code>lslv lvol_name</code>
VxVM	Display all volume information.	<code>vxprint -vt</code>

Table 27-15 Displaying volume and disk group information (*continued*)

Task type	Description	Example
VxVM	Display information about a specific volume.	<code>vxprint -ht vol_name</code>
VxVM	Display disk group information. Use <code>vxdisk</code> to display information about a specific disk group.	<code>vxdisk list</code> <code>vxprint -g disk_group</code> <code>vxdbg list</code> <code>vxdbg list disk_group</code>
LVM	Display information about physical volumes.	<code>lspv disk_name</code>
VxVM	Display information about Volume Manager volumes.	<code>vxinfo</code> or <code>vxprint</code>

[Table 27-16](#) shows the commands for removing volumes.

Table 27-16 Removing volumes

Task type	Description	Example
LVM	Remove a logical volume.	<code>rmlv lvol_name</code>
VxVM	Remove a volume.	<code>vxedit -g disk_group \ rm vol_name</code> <code>vxassist -g disk_group \ remove volume vol_name</code>

[Table 27-17](#) shows the commands for removing a group.

Table 27-17 Removing a group

Task type	Description	Example
LVM	Remove an entire volume group. Before attempting to remove the volume group, you must remove the logical volumes using <code>rmlv</code> , and all physical volumes except the last one using <code>reducevg</code>	<code>reducevg vol_grp last_disk...</code>

Table 27-17 Removing a group (*continued*)

Task type	Description	Example
VxVM	Destroy a disk group. You must unmount and stop any volumes in the disk group first.	<code>vxdg destroy disk_group</code>

[Table 27-18](#) shows the commands for creating mirrored volumes.

Table 27-18 Creating mirrored volumes

Task type	Description	Example
LVM	Create a mirrored logical volume.	<code>mklv -c 2 -y vol_name \ -t type vgmirror NumOfLPs \ [PVname...]</code>
VxVM	Create a mirrored volume/plex or add a mirror to an existing volume.	<code>vxassist -g group_name \ make vol_name length \ layout=mirror</code>

[Table 27-19](#) shows the commands for removing mirrors.

Table 27-19 Removing mirrors

Task type	Description	Example
LVM	Reduce a single/double mirrored logical volume to an unmirrored logical volume. Remove a mirrored logical volume.	<code>rmlvcopy mirr_lv #</code> Where # is the number of copies to remove. <code>rmlv LVname</code>
VxVM	<code>vxplex</code> removes mirrors or reduces the number of plexes/mirrors. <code>vxedit</code> removes a volume with the plexes associated with it.	<code>vxplex -g group_name \ -o rm dis plex_name</code> <code>vxedit -g group_name \ -rf rm vol_name</code>

[Table 27-20](#) shows the commands for increasing the number of mirrors.

Table 27-20 Increasing the number of mirrors

Task type	Description	Example
LVM	Increase the number of mirror copies.	<code>extendlv vol_name NumOfLPs</code>
VxVM	Add mirrors to a volume or increase the number of plexes.	<code>vxassist -g group_name \ mirror vol_name</code>

[Table 27-21](#) shows the commands for splitting a volume.

Table 27-21 Splitting a volume

Task type	Description	Example
LVM	Convert a mirrored logical volume into two logical volumes. Split a logical volume.	<code>splitlvcopy -y NewLVName \ OldLVName copies</code>
VxVM	Snapshot a volume and create a new volume.	<code>vxassist -g group_name \ snapstart vol_name</code> <code>vxassist -g group_name \ snapshot vol_name \ mir_vol_name</code>

[Table 27-22](#) shows the commands for moving a mirrored volume.

Table 27-22 Moving a mirrored volume

Task type	Description	Example
LVM	Move a mirrored logical volume from one disk to another.	Use the <code>migratepv</code> command
VxVM	Move a plex.	<code>vxplex -g group_name \ mv orig_plex new_plex</code>

[Table 27-23](#) shows the commands for synchronizing mirrored volumes.

Table 27-23 Synchronizing mirrored volumes

Task type	Description	Example
LVM	Synchronize a mirrored logical volume. Synchronize extents within a mirrored logical volume.	<code>syncvg -l lvol_name</code>
VxVM	Resynchronize operations for the given volumes.	<code>vxvol -g group_name\ resync volname</code>
VxVM	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.	<code>vxrecover -g group_name \ -s vol_name</code>

[Table 27-24](#) shows the commands for starting volumes.

Table 27-24 Starting volumes

Task type	Description	Example
LVM	Start a volume.	Use the <code>varyonvg</code> command
VxVM	Start a volume.	<code>vxrecover -g group_name \ -s vol_name</code> <code>vxvol -g group_name start \ vol_name</code>

[Table 27-25](#) shows the commands for stopping volumes.

Table 27-25 Stopping volumes

Task type	Description	Example
LVM	Stop a volume.	Use the <code>varyoffvg</code> command.
VxVM	Stop a volume.	<code>vxvol -g group_name stop \ vol_name</code>

[Table 27-26](#) shows the commands for replacing a disk.

Table 27-26 Replacing a disk

Task type	Description	Example
LVM	Replace a disk.	Use the <code>replacevg</code> command.
VxVM	Replace a disk.	Select the option <code>Replace a failed or removed disk of vxdiskadm</code> .

[Table 27-27](#) shows the commands for renaming a group.

Table 27-27 Renaming a group

Task type	Task description	Example
LVM	Rename a volume group.	Use the <code>varyonvg</code> command with the new name.
VxVM	Rename a disk group.	<code>vxchg -tC -n newdg_name</code>

[Table 27-28](#) shows the commands for renaming a volume.

Table 27-28 Renaming a volume

Task type	Task description	Example
LVM	Rename a logical volume.	<code>chlv -n</code>
VxVM	Rename a volume.	<code>vxedit -g disk_group \</code> <code>-v rename name newname</code> Update the <code>/etc/filesystems</code> file with the new name.

[Table 27-29](#) shows the commands for moving volumes off of a disk.

Table 27-29 Moving volumes off of a disk

Task type	Task description	Example
LVM	Move volumes off of a disk.	<code>migratepv</code>

Table 27-29 Moving volumes off of a disk (*continued*)

Task type	Task description	Example
VxVM	Move volumes off of a disk.	<p><code>vxevac</code></p> <p>In <code>vxdiskadm</code>, use the option <code>Move</code> volumes from a disk to move volumes.</p>

Tasks with no direct LVM equivalents

[Table 27-30](#) lists tasks which have no direct LVM equivalent.

Most of these tasks can be performed either with the Storage Foundation Manager (SFM) or the command line interface.

This list includes only common tasks, and is not exhaustive.

For more information, refer to the following documents:

- *Veritas Volume Manager Administrator's Guide*
- *Veritas Cross-platform Data Sharing Administrator's Guide*
- *Veritas Intelligent Storage Provisioning Administrator's Guide*
- *Veritas Storage Foundation Manager User's Guide*

Table 27-30 Additional VxVM tasks with no LVM equivalents

Task description	Example
Hot-relocation designates a disk as a hot-relocation spare and allows the system to automatically react to I/O failure by relocating redundant subdisks to other disks. The <code>vxunreloc</code> utility can be used to restore the system to the same configuration that existed before the disk failure.	<p><code>vxedit -g disk_group</code> <code>set spare=on disk_name</code></p> <p>Alternatively, use menu option <code>Mark</code> a disk as a spare for a disk group of <code>vxdiskadm</code> to perform this task.</p>
Offline a disk.	<p><code>vxdisk offline disk_name</code></p> <p>Alternatively, use menu option <code>Disable</code> (offline) a disk device of <code>vxdiskadm</code> to perform this task.</p>

Table 27-30 Additional VxVM tasks with no LVM equivalents (*continued*)

Task description	Example
Online a disk.	<code>vxdisk online disk_name</code> Select menu option Enable (online) a disk device of vxdiskadm.
Evacuate a disk.	<code>vxevac -g disk_group medianame \ new_medianame</code>
Recover volumes on a disk.	<code>vxrecover -g disk_group \ vol_name medianame</code>
Display a DMP node.	<code>vxdisk list meta_device</code>
Prepare a volume for DRL or instant snapshot operations.	<code>vxsnap -g disk_group prepare vol_name</code>
Create a full-sized instant snapshot copy of a volume.	<code>vxsnap -g disk_group make \ source=vol_name/snapvol=temp_vol_name</code>
Create a space-optimized snapshot copy of a volume.	<code>vxsnap -g disk_group make \ source=vol_name/newvol= temp_vol_name/ cache=cache_object</code>
Recover a volume.	<code>vxrecover -g disk_group volume \ medianame</code> <code>vxmend fix clean plex_name</code>
Repair a mirror.	<code>vxplex att plex_name</code>
Disable a mirror.	<code>vxplex det plex_name</code>
Remove a log from a volume.	<code>vxassist -g disk_group remove \ log vol_name</code>
Move a subdisk.	<code>vxsd -g disk_group mv \ old_subdisk new_subdisk</code>

Table 27-30 Additional VxVM tasks with no LVM equivalents (*continued*)

Task description	Example
Decrease the disk space allocated to a logical volume with the <code>shrinkto</code> or <code>shrinkby</code> parameters. Note: Make sure you shrink the file system before shrinking the volume.	<pre>vxassist -g disk_group \ shrinkto vol_name new_size</pre>
Remove one or more volumes from a volume group.	<pre>vxedit -g disk_group -rf rm \ vol_name</pre> <pre>vxassist -g disk_group remove \ volume vol_name</pre>
Refresh the contents of an instant snapshot from a source volume.	<pre>vxsnap -g disk_group refresh \ snap_volume source=vol_name</pre>
Reattach the plexes of a full-sized instant snapshot.	<pre>vxsnap -g disk_group reattach \ snap_volume source=vol_name</pre>
Move the contents of a subdisk onto new subdisks and replace the old sub disk with the new subdisks for any associations.	<pre>vxsd -g disk_group mv</pre>
Restore disk group configuration.	<pre>vxconfigrestore</pre>

LVM features not supported in VxVM

Some of the features in LVM are not supported in the current release of VxVM.

[Table 27-31](#) shows the unsupported LVM features, and possible workarounds in VxVM.

Table 27-31 LVM features and VxVM equivalents

LVM feature	VxVM equivalent
Physical volume groups	VxVM has no equivalent feature. The disk group feature of VxVM combines the logical volume group (VG) and physical volume group (PVG) of LVM.

Table 27-31 LVM features and VxVM equivalents (*continued*)

LVM feature	VxVM equivalent
Bad media block relocation	VxVM relocates whole subdisks. Smaller granularity relocation is not supported. The bad block reallocation feature does not exist in VxVM because the vectoring of bad blocks is now done by most hardware.

System Management Interface Tool (SMIT)

About the AIX System Management Interface Tool

This chapter describes how to use the AIX System Management Interface Tool (SMIT) to administer VxVM, and how to launch SMIT from the command line.

Note: Only privileged users can run SMIT.

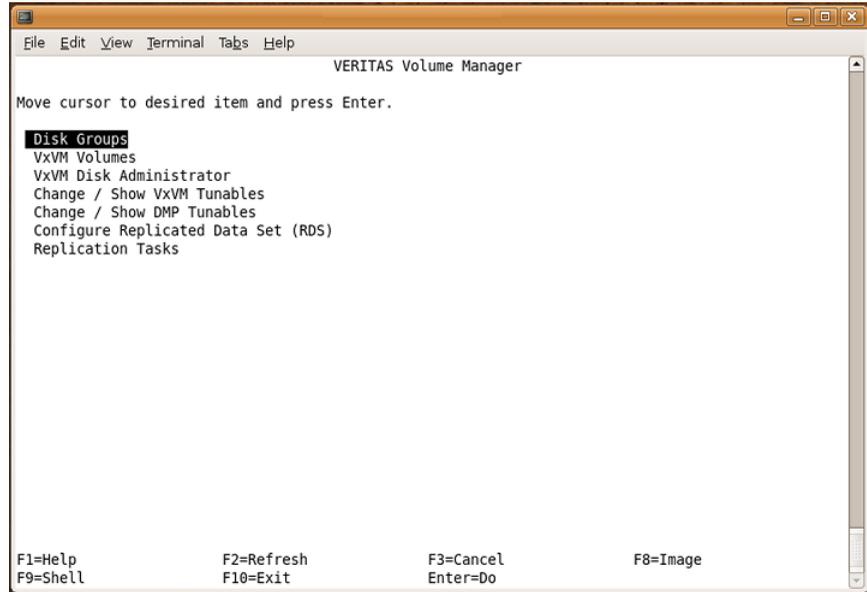
Launching SMIT

Launch the SMIT interface to VxVM by entering the command:

```
$ smit vxvm
```

[Figure 27-1](#) shows the top-level menu.

Figure 27-1 VxVM main menu



From this menu you can perform administrative tasks on VxVM components such as disk groups, disks and volumes, including the following tasks:

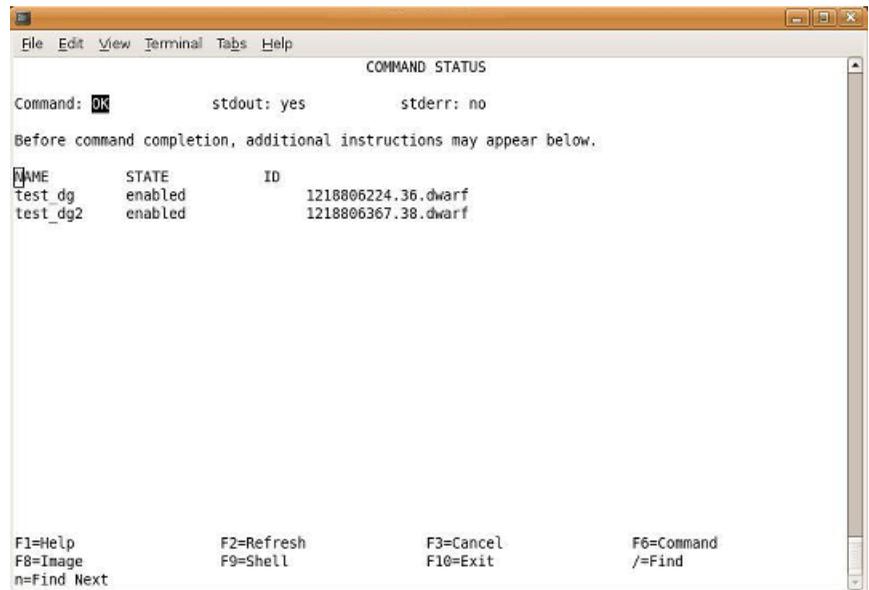
- Operations on VxVM disk groups (list/add/remove/modify/import/deport)
- Operations on VxVM Volumes (list/add/remove/change/snapshot)
- A subset of the operations that are available through the `vxdiskadm` command
- Changing VxVM tunables
- Changing DMP tunables
- Replication tasks

Administering disk groups in SMIT

Figure 27-2 shows a listing of disk groups in SMIT.

To view this listing, select from the Disks and File Systems area, then select VxVM Disk Administrator>List all Disk Groups.

Figure 27-2 Listing disk groups



The Disk Groups interface lists the VxVM disk groups on the system. In this example, there are two disk groups, `test_dg` and `test_dg2`.

Administering disk devices in SMIT

Figure 27-3 shows how to list all disks on the system.

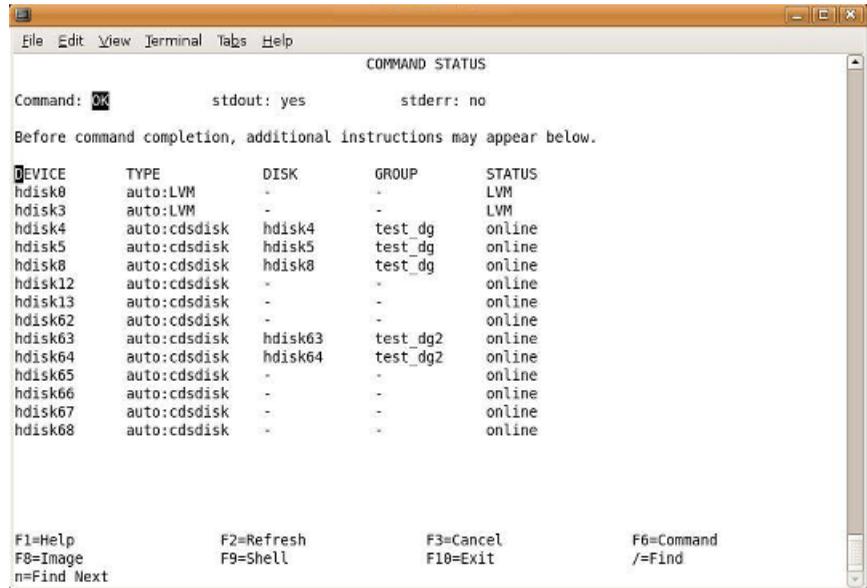
To list all disks, select VxVM Disk Administrator>List all disks in the system.

When VxVM is installed on the system, SMIT includes a “STATUS” column to indicate whether a disk is under LVM or VxVM control, or whether it is unused.

If a VxVM disk is online and part of a disk group, the disk group name is listed under the “Group” column. If a VxVM disk is initialized, but not yet part of a disk group, the entries in the “Disk” and “Group” columns are blank.

In the example, devices `hdisk4`, `hdisk5` and `hdisk8` are part of the `test_dg` disk group, devices `hdisk63` through `hdisk64` are part of the `test_dg2` disk group, and device `hdisk0` and `hdisk3` are LVM disks.

Figure 27-3 Listing disks

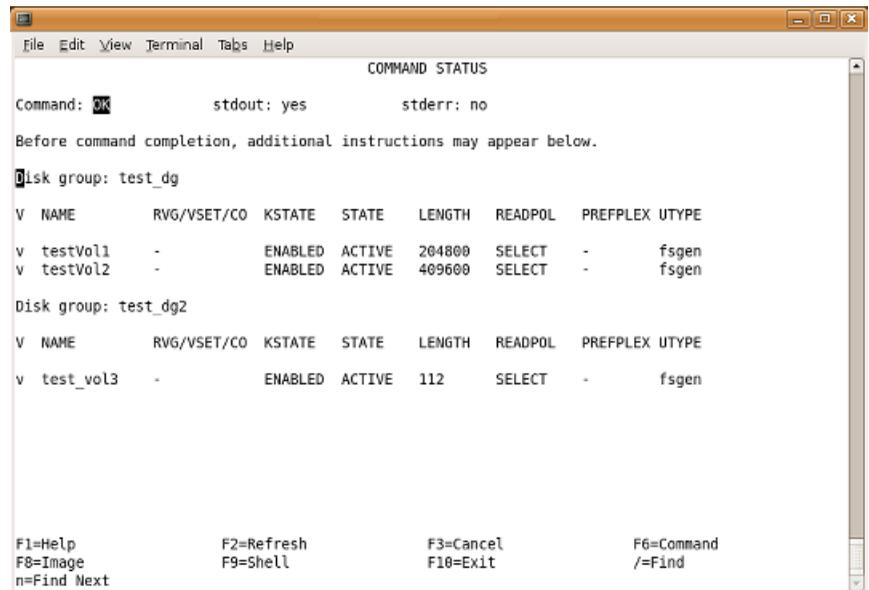


Administering volumes in SMIT

To list logical volumes in SMIT, select VxVM Volumes from the main menu, and then select List VxVM Volumes in all or specific Disk Groups. You will then be prompted to enter the name of a disk group.

Figure 27-4 displays the result of entering All (the default).

Figure 27-4 Listing volumes in a system



This figure shows that the system consists of two disk groups, each of which in turn consists of one or more volumes.

Full information on the displayed output can be found in the following document:

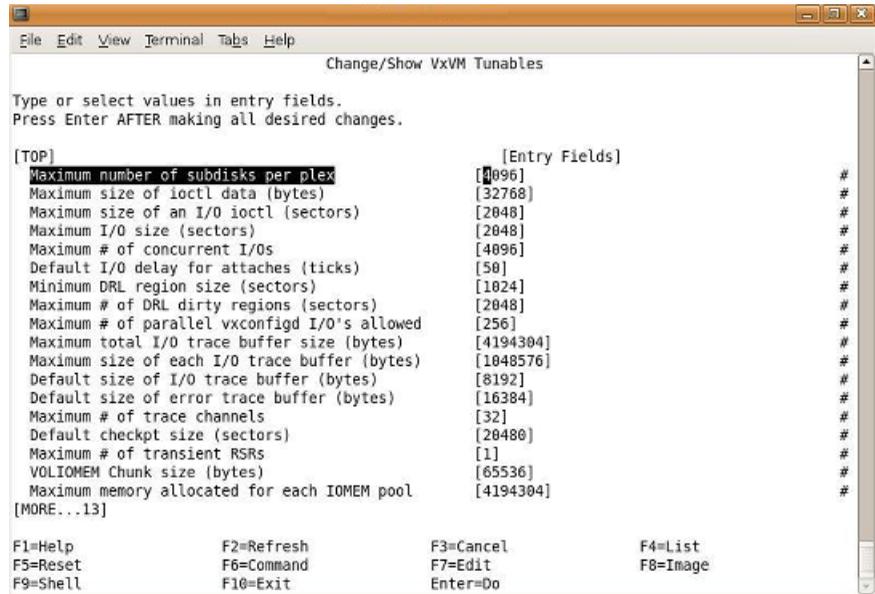
Veritas Volume Manager Administrator's Guide.

Administering VxVM tunables in SMIT

The following example of a SMIT interface is a screen that allows you to list and amend VxVM tunable parameters.

[Figure 27-5](#) shows the screen displayed, from the main menu, by selecting Change/Show VxVM Tunables.

Figure 27-5 Displaying tunable parameters and their current values



By using this display, you can configure your system to meet your own particular requirements.

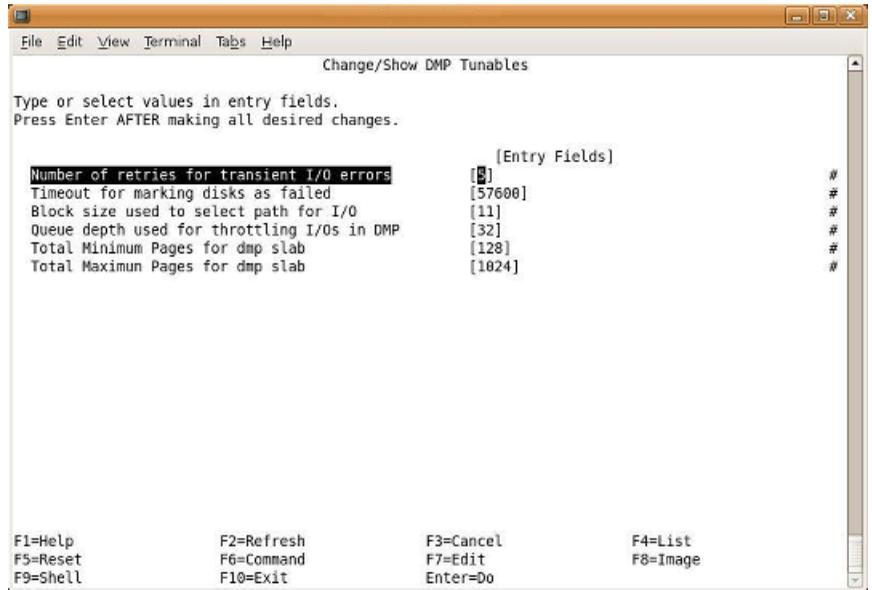
For more information on Veritas Volume Manager tasks, refer to the Veritas Volume Manager documentation.

Administering DMP tunables in SMIT

The following example of a SMIT interface is a screen that allows you to list and amend tunable parameters used by the Dynamic Multipathing (DMP) feature of VxVM.

Figure 27-5 shows the screen displayed, from the main menu, by selecting Change/Show DMP Tunables.

Figure 27-6 Displaying tunable parameters and their current values



By using this display, you can configure your system to meet your own particular requirements.

For more information on DMP, refer to the Veritas Volume Manager documentation.

Online migration of a native file system to the VxFS file system

This chapter includes the following topics:

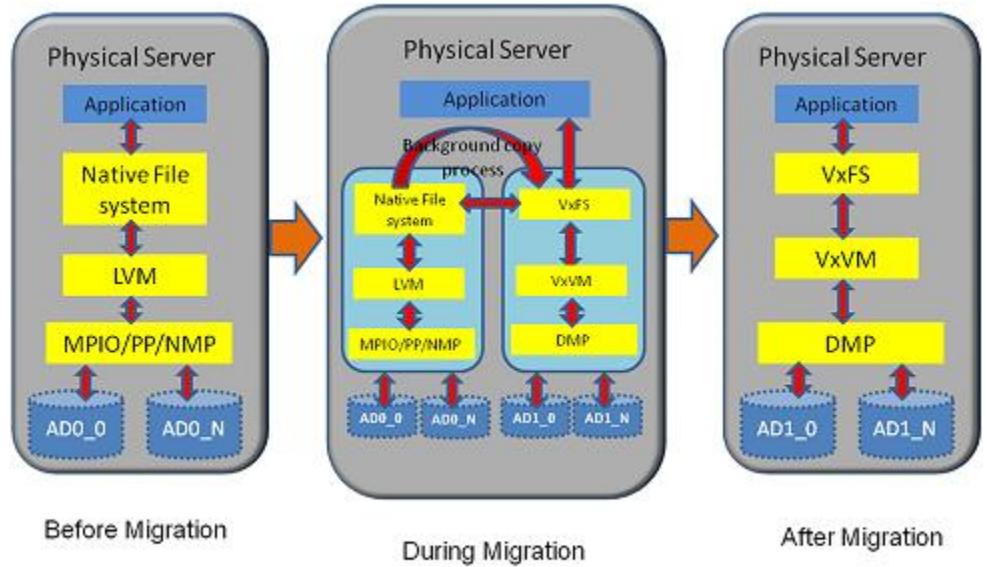
- [About online migration](#)
- [Administrative interface for online migration](#)
- [Migrating a native file system to the VxFS file system](#)
- [Backing out an online migration operation](#)
- [VxFS features not available during online migration](#)

About online migration

The online migration feature provides a method to migrate a native file system to the VxFS file system. The native file system is referred to as source file system and the VxFS file system as referred to as the target file system. The migration takes minimum amounts of clearly bounded, easy to schedule downtime. Online migration is not an in-place conversion and requires a separate storage. During online migration the application remains online and the native file system data is copied over to the VxFS file system. Both of the file systems are kept in sync during migration. This makes online migration back-out and recovery seamless. The online migration tool also provides an option to throttle the background copy operation to speed up or slow down the migration based on your production needs.

[Figure 28-1](#) illustrates the overall migration process.

Figure 28-1 Migration process



You can migrate JFS and JFS2 file systems.

Administrative interface for online migration

Online migration of a native file system to the VxFS file system can be started using the `fsmigadm VxFS` administrative command.

Table 28-1 describes the `fsmigadm` keywords.

Table 28-1

Keyword	Usage
<code>analyze</code>	Analyzes the source file system that is to be converted to VxFS and generates an analysis report.
<code>start</code>	Starts the migration.
<code>list</code>	Lists all ongoing migrations.
<code>status</code>	Shows a detailed status of the migration for the specified file system, or all file systems under migration.
<code>throttle</code>	Throttles the background copy operation.
<code>pause</code>	Pauses the background copy operation for one or more of migrations.

Table 28-1 (continued)

Keyword	Usage
<code>resume</code>	Resumes the background copy operation if the operation was paused or the background copy operation was killed before the migration completed.
<code>commit</code>	Commits the migration.
<code>abort</code>	Aborts the migration.

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

Migrating a native file system to the VxFS file system

The following procedure migrates a native file system to the VxFS file system.

Note: You cannot unmount the target (VxFS) file system nor the source file system after you start the migration. Only the `commit` or `abort` operation can unmount the target file system. Do not force unmount the source file system; use the `abort` operation to stop the migration and unmount the source file system.

To migrate a native file system to the VxFS file system

- 1 Install Veritas Storage Foundation on the physical application host.
See the *Veritas Storage Foundation Installation Guide*.
- 2 Add new storage to the physical application host on which you will configure Veritas Volume Manager (VxVM).
- 3 Create a VxVM volume according to the your desired configuration on the newly added storage. The volume size cannot be less than source file system size.
- 4 Mount the source file system if the file system is not mounted already.
- 5 Run the `fsmigadm analyze` command and ensure that all checks pass:

```
# fsmigadm analyze /dev/vx/dsk/dg/vol1 /mnt1
```

- 6 If the application is online, then shut down the application.
- 7 Start the migration by running `fsmigadm start`:

```
# fsmigadm start /dev/vx/dsk/vol1 /mnt1
```

The `fsmigadm` command performs the following tasks:

- Unmounts the source file system.
- Creates a VxFS file system using the `mkfs` command on the new storage provided, specifying the same block size (*bsize*) as the source file system.
- Mounts the target file system.
- Mounts the source file system inside the target file system, as `/mnt1/lost+found/srcfs`.

You can perform various operations during the migration.

- You can get the status of the migration using the `fsmigadm status` command:

```
# fsmigadm status /mnt1
```

- You can speed up or slow down the migration using the `fsmigadm throttle` command:

```
# fsmigadm throttle 9g /mnt1
```

- You can pause the migration using `fsmigadm pause` command:

```
# fsmigadm pause /mnt1
```

- You can resume the migration using the `fsmigadm resume` command:

```
# fsmigadm resume /mnt1
```

The application can remain online throughout the entire migration operation. When the migration operation completes, you are alerted via the system log.

Both of the file systems are kept up-to-date until the migration is committed.

- 8 Bring the application online.
- 9 Check the log file for any errors during migration. If you find any errors, you must copy the indicated files manually from the source file system after performing the commit operation.
- 10 Shutdown the application.

11 Commit the migration:

```
# fsmigadm commit /mnt1
```

The `fsmigadm` command unmounts the source file system, unmounts the target file system, then mounts the target file system.

12 Start the application on the Veritas Storage Foundation stack.

Backing out an online migration operation

The following procedure backs out an online migration operation of a native file system to the VxFS file system.

Note: As both source and target file system are kept in sync during migration, the application sometimes experiences performance degradation.

In the case of a system failure, if the migration operation completed before the system crashed, then you are able to use the VxFS file system.

To back out an online migration operation of a native file system to the VxFS file system

- 1 Shut down the application
- 2 Abort the migration:

```
# fsmigadm abort /mnt1
```

The source file system is mounted again.

- 3 Bring the application online.

VxFS features not available during online migration

The following features are not supported on a file system that you are migrating:

- Block clear (`blkclear`) mount option
- Cached Quick I/O
- Cross-platform data sharing (portable data containers)
- Data management application programming interface (DMAPI)
- File Change Log (FCL)
- File promotion (undelete)

- Fileset quotas
- Forced unmount
- Online resize
- Quick I/O
- Quotas
- Reverse name lookup
- SmartTier
- Snapshots
- Storage Checkpoints
- Veritas Storage Foundation Cluster File System (SFCFS)

The following commands are not supported on a file system that you are migrating:

- `fiostat`
- `fsadm`
- `tar`
- `vxdump`
- `vxfreeze`
- `vxrestore`
- `vxupgrade`

All of the listed features and commands become available after you commit the migration.

Note: Any source file system-specific features are not preserved on the target file system.

You cannot use online migration on a nested source mount point.

You cannot migrate from a VxFS file system nor an NFS file system to a VxFS file system.

Migrating data between platforms

This chapter includes the following topics:

- [Overview of CDS](#)
- [CDS disk format and disk groups](#)
- [Setting up your system](#)
- [Maintaining your system](#)
- [File system considerations](#)
- [Alignment value and block size](#)
- [Migrating a snapshot volume](#)

Overview of CDS

This section presents an overview of the Cross-Platform Data Sharing (CDS) feature of Symantec's Veritas Storage Foundation™ software. CDS provides you with a foundation for moving data between different systems within a heterogeneous environment. The machines may be running HP-UX, AIX, Linux or the Solaris™ operating system (OS), and they may all have direct access to physical devices holding data. CDS allows Symantec's Veritas products and applications to access data storage independently of the operating system platform, enabling them to work transparently in heterogeneous environments.

The Cross-Platform Data Sharing feature is also known as Portable Data Containers (PDC). For consistency, this document uses the name Cross-Platform Data Sharing throughout.

The following levels in the device hierarchy, from disk through file system, must provide support for CDS to be used:

End-user applications	Application level.
Veritas™ File System (VxFS)	File system level.
Veritas™ Volume Manager (VxVM)	Volume level.
Operating system	Device level.

CDS is a license-enabled feature that is supported at the disk group level by VxVM and at the file system level by VxFS.

CDS utilizes a new disk type (`auto:cdsdisk`). To effect data sharing, VxVM supports a new disk group attribute (`cds`) and also supports different OS block sizes.

Note: CDS allows data volumes and their contents to be easily migrated between heterogeneous systems. It does not enable concurrent access from different types of platform unless such access is supported at all levels that are required.

Shared data across platforms

While volumes can be exported across platforms, the data on the volumes can be shared only if data sharing is supported at the application level. That is, to make data sharing across platforms possible, it must be supported throughout the entire software stack.

For example, if a VxFS file system on a VxVM volume contains files comprising a database, then the following functionality applies:

- Disks can be recognized (as `cds` disks) across platforms.
- Disk groups can be imported across platforms.
- The file system can be mounted on different platforms.

However, it is very likely that, because of the inherent characteristics of databases, you may not be able to start up and use the database on a platform different from the one on which it was created. (A notable exception is Oracle 10g's Cross-Platform Transportable Tablespace feature.)

An example is where an executable file, compiled on one platform, can be accessed across platforms (using CDS), but may not be executable on a different platform.

Note: You do not need a file system in the stack if the operating system provides access to raw disks and volumes, and the application can utilize them. Databases and other applications can have their data components built on top of raw volumes without having a file system to store their data files.

Disk drive sector size

Sector size is an attribute of a disk drive (or SCSI LUN for an array-type device), which is set when the drive is formatted. Sectors are the smallest addressable unit of storage on the drive, and are the units in which the device performs I/O. The sector size is significant because it defines the atomic I/O size at the device level. Any multi-sector writes which VxVM submits to the device driver are not guaranteed to be atomic (by the SCSI subsystem) in the case of system failure.

Block size issues

The block size is a platform-dependent value that is greater than or equal to the sector size. Each platform accesses the disk on block boundaries and in quantities that are multiples of the block size.

Data that is created on one platform, and then accessed by a platform of a different block size, can suffer from the following problems:

- Addressing issues
- The data may not have been created on a block boundary compatible with that used by the accessing platform.
 - The accessing platform cannot address the start of the data.

Bleed-over issues The size of the data written may not be an exact multiple of the block size used by the accessing platform. Therefore the accessing platform cannot constrain its I/O within the boundaries of the data on disk.

Operating system data

Some operating systems (OS) require OS-specific data on disks in order to recognize and control access to the disk.

CDS disk format and disk groups

This section provides additional information about CDS disk format and CDS disk groups.

CDS disk access and format

For a disk to be accessible by multiple platforms, the disk must be consistently recognized by the platforms, and all platforms must be capable of performing I/O on the disk. CDS disks contain specific content at specific locations to identify or control access to the disk on different platforms. The same content and location are used on all CDS disks, independent of the platform on which the disks are initialized.

In order for a disk to be initialized as, or converted to a CDS disk, it must satisfy the following requirements:

- Must be a SCSI disk
- Must be the entire physical disk (LUN)
- Only one volume manager (such as VxVM) can manage a physical disk (LUN)
- There can be no disk partition (slice) which is defined, but which is not configured on the disk
- Cannot contain a volume whose use-type is either `root` or `swap` (for example, it cannot be a boot disk)

The CDS conversion utility, `vxcdsconvert`, is provided to convert non-CDS VM disk formats to CDS disks, and disk groups with a version number less than 110 to disk groups that support CDS disks.

See [“Converting non-CDS disks to CDS disks”](#) on page 463.

CDS disk types

The CDS disk format, `cdsdisk`, is recognized by all VxVM platforms (including Windows). The `cdsdisk` disk format is the default for all newly-created VM disks unless overridden in a defaults file. The `vxcdsconvert` utility is provided to convert other disk formats and types to CDS.

See [“Defaults files”](#) on page 467.

Note: Disks with format `cdsdisk` can only be added to disk groups with version 110 or later.

Private and public regions

A VM disk usually has a private and a public region.

The private region is a small area on the disk where VxVM configuration information is stored, such as a disk header label, configuration records for VxVM objects (such as volumes, plexes and subdisks), and an intent log for the

configuration database. The default private region size is 32MB, which is large enough to record the details of several thousand VxVM objects in a disk group.

The public region covers the remainder of the disk, and is used for the allocation of storage space to subdisks.

The private and public regions are aligned and sized in multiples of 8K to permit the operation of CDS. The alignment of VxVM objects within the public region is controlled by the disk group alignment attribute. The value of the disk group alignment attribute must also be 8k to permit the operation of CDS.

Note: With other (non-CDS) VxVM disk formats, the private and public regions are aligned to the platform-specific OS block size.

Disk access type auto

The disk access (DA) disk type auto supports multiple disk formats, including cdsdisk, which is supported across all platforms. It is associated with the DA records created by the VxVM auto-configuration mode. Disk type auto automatically determines which format is on the disk.

Platform block

The platform block resides on disk sector 0, and contains data specific to the operating system for the platforms. It is necessary for proper interaction with each of those platforms. The platform block allows a disk to perform as if it was initialized by each of the specific platforms.

AIX coexistence label

The AIX coexistence label resides on the disk, and identifies the disk to the AIX logical volume manager (LVM) as being controlled by VxVM.

HP-UX coexistence label

The HP-UX coexistence label resides on the disk, and identifies the disk to the HP logical volume manager (LVM) as being controlled by VxVM.

VxVM ID block

The VxVM ID block resides on the disk, and indicates the disk is under VxVM control. It provides dynamic VxVM private region location and other information.

CDS disk groups

A CDS disk group allows cross-platform data sharing of VxVM objects, so that data written on one of the supported platforms may be accessed on any other

supported platform. A CDS disk group is composed only of CDS disks (VM disks with the disk format `cdsdisk`), and is only available for disk group version 110 and greater.

Starting with disk group version 160, CDS disk groups can support disks of greater than 1 TB.

Note: The CDS conversion utility, `vxcdsconvert`, is provided to convert non-CDS VM disk formats to CDS disks, and disk groups with a version number less than 110 to disk groups that support CDS disks.

See [“Converting non-CDS disks to CDS disks”](#) on page 463.

All VxVM objects in a CDS disk group are aligned and sized so that any system can access the object using its own representation of an I/O block. The CDS disk group uses a platform-independent alignment value to support system block sizes of up to 8K.

See [“Disk group alignment”](#) on page 457.

CDS disk groups can be used in the following ways:

- Initialized on one system and then used “as-is” by VxVM on a system employing a different type of platform.
- Imported (in a serial fashion) by Linux, Solaris, AIX, and HP-UX systems.
- Imported as private disk groups, or shared disk groups (by CVM).

You cannot include the following disks or volumes in a CDS disk group:

- Volumes of usage type `root` and `swap`. You cannot use CDS to share boot devices.
- Encapsulated disks.

Note: On Solaris and Linux systems, the process of disk encapsulation places the slices or partitions on a disk (which may contain data or file systems) under VxVM control. On AIX and HP-UX systems, LVM volumes may similarly be converted to VxVM volumes.

Device quotas

Device quotas limit the number of objects in the disk group which create associated device nodes in the file system. Device quotas are useful for disk groups which need to be transferred between Linux with a pre-2.6 kernel and other supported platforms. Prior to the 2.6 kernel, Linux supported only 256 minor devices per major device.

You can limit the number of devices that can be created in a given CDS disk group by setting the device quota.

See [“Setting the maximum number of devices for CDS disk groups”](#) on page 474.

When you create a device, an error is returned if the number of devices would exceed the device quota. You then either need to increase the quota, or remove some objects using device numbers, before the device can be created.

See [“Displaying the maximum number of devices in a CDS disk group”](#) on page 478.

Minor device numbers

Importing a disk group will fail if it will exceed the maximum devices for that platform.

Note: There is a large disparity between the maximum number of devices allowed for devices on the Linux platform with a pre-2.6 kernel, and that for other supported platforms.

Non-CDS disk groups

Any version 110 (or greater) disk group (DG) can contain both CDS and non-CDS disks. However, only version 110 (or greater) disk groups composed entirely of CDS disks have the ability to be shared across platforms. Whether or not that ability has been enabled is controlled by the `cds` attribute of the disk group. Enabling this attribute causes a non-CDS disk group to become a CDS disk group.

Although a non-CDS disk group can contain a mixture of CDS and non-CDS disks having dissimilar private region alignment characteristics, its disk group alignment will still direct how all subdisks are created.

Disk group alignment

One of the attributes of the disk group is the block alignment, which represents the largest block size supported by the disk group.

The alignment constrains the following attributes of the objects within a disk group:

- Subdisk offset
- Subdisk length
- Plex offset
- Volume length
- Log length

- **Stripe width**

The offset value specifies how an object is positioned on a drive.

The disk group alignment is assigned at disk group creation time.

See [“Disk group tasks”](#) on page 471.

Alignment values

The disk group block alignment has two values: 1 block or 8k (8 kilobytes).

All CDS disk groups must have an alignment value of 8k.

All disk group versions before version 110 have an alignment value of 1 block, and they retain this value if they are upgraded to version 110 or later.

A disk group that is not a CDS disk group, and which has a version of 110 and later, can have an alignment value of either 1 block or 8k.

The alignment for all newly initialized disk groups in VxVM 4.0 and later releases is 8k. This value, which is used when creating the disk group, cannot be changed. However, the disk group alignment can be subsequently changed.

See [“Changing the alignment of a non-CDS disk group”](#) on page 472.

Note: The default usage of `vxassist` is to set the `layout=diskalign` attribute on all platforms. The `layout` attribute is ignored on 8K-aligned disk groups, which means that scripts relying on the default may fail.

Dirty region log alignment

The location and size of each map within a dirty region log (DRL) must not violate the disk group alignment for the disk group (containing the volume to which the DRL is associated). This means that the region size and alignment of each DRL map must be a multiple of the disk group alignment, which for CDS disk groups is 8K. (Features utilizing the region size can impose additional minimums and size increments over and above this restriction, but cannot violate it.)

In a version 110 disk group, a traditional DRL volume has the following region requirements:

- Minimum region size of 512K
- Incremental region size of 64K

In a version 110 disk group, a version 20 DCO volume has the following region requirements:

- Minimum region size of 16K

- Incremental region size of 8K

Note: The map layout within a Data Change Object (DCO) volume changed with the release of VxVM 4.0 to version 20. This can accommodate both FastResync and DRL maps within the DCO volume. The original version 0 layout for DCO volumes only accommodates FastResync maps.

Object alignment during volume creation

For CDS disk groups, VxVM objects that are used in volume creation are automatically aligned to 8K. For non-CDS disk groups, the `vxassist` attribute, `dgalignment_checking`, controls how the command handles attributes that are subject to disk group alignment restrictions. If set to `strict`, the volume length and values of attributes must be integer multiples of the disk group alignment value, or the command fails and an error message is displayed. If set to `round` (default), attribute values are rounded up as required. If this attribute is not specified on the command-line or in a defaults file, the default value of `round` is used.

The `diskalign` and `nodiskalign` attributes of `vxassist`, which control whether subdisks are aligned on cylinder boundaries, is honored only for non-CDS disk groups whose alignment value is set to 1.

Setting up your system

In order to migrate data between platforms using CDS, set up your system to use CDS disks and CDS disk groups. The CDS license must be enabled. You can use the default files to configure the appropriate settings for CDS disks and disk groups.

[Table 29-1](#) describes the tasks for setting up your system to use CDS.

Table 29-1 Setting up CDS disks and CDS disk groups

Task	Procedures
Create the CDS disks.	<p>You can create a CDS disk in one of the following ways:</p> <ul style="list-style-type: none"> ■ Creating CDS disks from uninitialized disks See “Creating CDS disks from uninitialized disks” on page 460. ■ Creating CDS disks from initialized VxVM disks See “Creating CDS disks from initialized VxVM disks” on page 461. ■ Converting non-CDS disks to CDS disks See “Converting non-CDS disks to CDS disks” on page 463.
Create the CDS disk groups.	<p>You can create a CDS disk group in one of the following ways:</p> <ul style="list-style-type: none"> ■ Creating CDS disk groups See “Creating CDS disk groups” on page 462. ■ Converting a non-CDS disk group to a CDS disk group See “Converting a non-CDS disk group to a CDS disk group” on page 464.
Verify the CDS license.	Verifying licensing See “Verifying licensing” on page 466.
Verify the system defaults related to CDS.	Defaults files See “Defaults files” on page 467.

Creating CDS disks from uninitialized disks

You can create a CDS disk from an uninitialized disk by using one of the following methods:

- [Creating CDS disks by using vxdisksetup](#)
- [Creating CDS disks by using vxdiskadm](#)

Creating CDS disks by using vxdisksetup

To create a CDS disk by using the vxdisksetup command

- Type the following command:

```
# vxdisksetup -i disk [format=disk_format]
```

The format defaults to `cdsdisk` unless this is overridden by the `/etc/default/vxdisk` file, or by specifying the disk format as an argument to the `format` attribute.

See “[Defaults files](#)” on page 467.

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

Creating CDS disks by using vxdiskadm

To create a CDS disk by using the `vxdiskadm` command

- Run the `vxdiskadm` command, and select the “Add or initialize one or more disks” item from the main menu. You are prompted to specify the format.

Warning: On CDS disks, the CDS information occupies the first sector of that disk, and there is no `fdisk` partition information. Attempting to create an `fdisk` partition (for example, by using the `fdisk` or `format` commands) erases the CDS information, and can cause data corruption.

Creating CDS disks from initialized VxVM disks

How you create a CDS disk depends on the current state of the disk, as follows:

- [Creating a CDS disk from a disk that is not in a disk group](#)
- [Creating a CDS disk from a disk that is already in a disk group](#)

Creating a CDS disk from a disk that is not in a disk group

To create a CDS disk from a disk that is not in a disk group

- 1 Run the following command to remove the VM disk format for the disk:

```
# vxdiskunsetup disk
```

This is necessary as non-auto types cannot be reinitialized by `vxdisksetup`.

- 2 If the disk is listed in the `/etc/vx/darecs` file, remove its disk access (DA) record using the command:

```
# vxdisk rm disk
```

(Disk access records that cannot be configured by scanning the disks are stored in an ordinary file, `/etc/vx/darecs`, in the root file system. Refer to the `vxintro(1M)` manual page for more information.)

- 3 Rescan for the disk using this command:

```
# vxdisk scandisks
```

- 4 Type this command to set up the disk:

```
# vxdisksetup -i disk
```

Creating a CDS disk from a disk that is already in a disk group

To create a CDS disk from a disk that is already in a disk group

- Run the `vxcdsconvert` command.
See [“Converting non-CDS disks to CDS disks”](#) on page 463.

Creating CDS disk groups

You can create a CDS disk group in the following ways:

- [Creating a CDS disk group by using `vxdg init`](#)
- [Creating a CDS disk group by using `vxdiskadm`](#)

Creating a CDS disk group by using `vxdg init`

Note: The disk group version must be 110 or greater.

To create a CDS disk group by using the `vxdg init` command

- Type the following command:

```
# vxdg init diskgroup disklist [cds={on|off}]
```

The format defaults to a CDS disk group, unless this is overridden by the `/etc/default/vxdg` file, or by specifying the `cds` argument.

See the `vxdg(1M)` manual page for more information.

Creating a CDS disk group by using `vxdiskadm`

You cannot create a CDS disk group when encapsulating an existing disk, or when converting an LVM volume.

When initializing a disk, if the target disk group is an existing CDS disk group, `vxdiskadm` will only allow the disk to be initialized as a CDS disk. If the target disk group is a non-CDS disk group, the disk can be initialized as either a CDS disk or a non-CDS disk.

If you use the `vxdiskadm` command to initialize a disk into an existing CDS disk group, the disk must be added with the `cdsdisk` format.

The CDS attribute for the disk group remains unchanged by this procedure.

To create a CDS disk group by using the `vxdiskadm` command

- Run the `vxdiskadm` command, and select the “Add or initialize one or more disks” item from the main menu. Specify that the disk group should be a CDS disk group when prompted.

Converting non-CDS disks to CDS disks

Note: The disks must be of type of `auto` in order to be re-initialized as CDS disks.

To convert non-CDS disks to CDS disks

- 1 If the conversion is not going to be performed on-line (that is, while access to the disk group continues), stop any applications that are accessing the disks.
- 2 Type one of the following forms of the CDS conversion utility (`vxcdsconvert`) to convert non-CDS disks to CDS disks.

```
# vxcdsconvert -g diskgroup [-A] [-d defaults_file] \  
  [-o novolstop] disk_name [attribute=value] ...  
# vxcdsconvert -g diskgroup [-A] [-d defaults_file] \  
  [-o novolstop] alldisks [attribute=value] ...
```

The `alldisks` and `disk` keywords have the following effect

<code>alldisks</code>	Converts all non-CDS disks in the disk group into CDS disks.
<code>disk</code>	Specifies a single disk for conversion. You would use this option under the following circumstances: <ul style="list-style-type: none">■ If a disk in the non-CDS disk group has cross-platform exposure, you may want other VxVM nodes to recognize the disk, but not to assume that it is available for initialization.■ If the native Logical Volume Manager (LVM) that is provided by the operating system needs to recognize CDS disks, but it is not required to initialize or manage these disks.■ Your intention is to move the disk into an existing CDS disk group.

The conversion involves evacuating objects from the disk, reinitializing the disk, and relocating objects back to disk. You can specify the `-o novolstop` option to perform the conversion on-line (that is, while access to the disk continues). If the `-o novolstop` option is not specified, stop any applications that are accessing the disks, and perform the conversion off-line.

Warning: Specifying the `-o novolstop` option can greatly increase the amount of time that is required to perform conversion.

Before you use the `vxcdsconvert` command, make sure you understand its options, attributes, and keywords.

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

Converting a non-CDS disk group to a CDS disk group

To convert a non-CDS disk group to a CDS disk group

- 1 If the disk group contains one or more disks that you do not want to convert to CDS disks, use the `vxdbg move` or `vxdbg split` command to move the disks out of the disk group.
- 2 The disk group to be converted must have the following characteristics:
 - No dissociated or disabled objects.
 - No sparse plexes.
 - No volumes requiring recovery.
 - No volumes with pending snapshot operations.
 - No objects in an error state.

To verify whether a non-CDS disk group can be converted to a CDS disk group, type the following command:

```
# vxcdsconvert -g diskgroup -A group
```

- 3 If the disk group does not have a CDS-compatible disk group alignment, the objects in the disk group must be relayed out with a CDS-compatible alignment.
- 4 If the conversion is not going to be performed on-line (that is, while access to the disk group continues), stop any applications that are accessing the disks.
- 5 Type one of the following forms of the CDS conversion utility (`vxcdsconvert`) to convert a non-CDS disk group to a CDS disk group.

```
# vxcdsconvert -g diskgroup [-A] [-d defaults_file] \  
  [-o novolstop] alignment [attribute=value] ...  
# vxcdsconvert -g diskgroup [-A] [-d defaults_file] \  
  [-o novolstop] group [attribute=value] ...
```

The `alignment` and `group` keywords have the following effect:

<code>alignment</code>	Specifies alignment conversion where disks are not converted, and an object layout is performed on the disk group. A successful completion results in an 8K-aligned disk group. You might consider this option, rather than converting the entire disk group, if you want to reduce the amount of work to be done for a later full conversion to CDS disk group.
<code>group</code>	Specifies group conversion of all non-CDS disks in the disk group before relaying out objects in the disk group.

The conversion involves evacuating objects from the disk, reinitializing the disk, and relocating objects back to disk. You can specify the `-o novolstop` option to perform the conversion on-line (that is, while access to the disk group continues). If the `-o novolstop` option is not specified, stop any applications that are accessing the disks, and perform the conversion off-line.

Warning: Specifying the `-o novolstop` option can greatly increase the amount of time that is required to perform conversion.

Conversion has the following side effects:

- Non-CDS disk groups are upgraded by using the `vxvg upgrade` command. If the disk group was originally created by the conversion of an LVM volume group (VG), rolling back to the original LVM VG is not possible. If you decide to go through with the conversion, the rollback records for the disk group will be removed, so that an accidental rollback to an LVM VG cannot be done.
- Stopped, but startable volumes, are started for the duration of the conversion .
- Any volumes or other objects in the disk group that were created with the `layout=diskalign` attribute specified can no longer be disk aligned.
- Encapsulated disks may lose the ability to be unencapsulated.
- Performance may be degraded because data may have migrated to different regions of a disk, or to different disks.

In the following example, the disk group, `mydg`, and all its disks are converted to CDS while its volumes are still online:

```
# vxcdsconvert -g mydg -o novolstop group \  
  move_subdisks_ok=yes evac_subdisks_ok=yes \  
  evac_disk_list=disk11,disk12,disk13,disk14
```

The `evac_disk_list` attribute specifies a list of disks (`disk11` through `disk14`) to which subdisks can be evacuated if required.

Before you use the `vxcdsconvert` command, make sure you understand its options, attributes, and keywords.

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

Verifying licensing

The ability to create or import a CDS disk group is controlled by a CDS license. CDS licenses are included as part of the Veritas Storage Foundation license.

To verify the CDS enabling license

- Type the following command:

```
# vxlicrep
```

Verify the following line in the output:

```
Cross-platform Data Sharing = Enabled
```

Defaults files

The following system defaults files in the `/etc/default` directory are used to specify the alignment of VxVM objects, the initialization or encapsulation of VM disks, the conversion of LVM disks, and the conversion of disk groups and their disks to the CDS-compatible format

`vxassist` Specifies default values for the following parameters to the `vxcdsconvert` command that have an effect on the alignment of VxVM objects: `dgalign_checking`, `diskalign`, and `nodiskalign`.

See “Object alignment during volume creation” on page 459.

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

`vxcdsconvert` Specifies default values for the following parameters to the `vxcdsconvert` command: `evac_disk_list`, `evac_subdisks_ok`, `min_split_size`, `move_subdisks_ok`, `privlen`, and `split_subdisks_ok`.

The following is a sample `vxcdsconvert` defaults file:

```
evac_subdisks_ok=no
min_split_size=64k
move_subdisks_ok=yes
privlen=2048
split_subdisks_ok=move
```

An alternate defaults file can be specified by using the `-d` option with the `vxcdsconvert` command.

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

`vx dg` Specifies default values for the `cds`, `default_activation_mode` and `enable_activation` parameters to the `vx dg` command. The `default_activation_mode` and `enable_activation` parameters are only used with shared disk groups in a cluster.

The following is a sample `vx dg` defaults file:

```
cds=on
```

See the `vx dg(1M)` manual page.

`vxdisk` Specifies default values for the `format` and `privlen` parameters to the `vxdisk` and `vxdisksetup` commands. These commands are used when disks are initialized by VxVM for the first time. They are also called implicitly by the `vxdiskadm` command and the Storage Foundation Manager (SFM) GUI.

The following is a sample `vxdisk` defaults file:

```
format=cdsdisk
privlen=2048
```

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

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

`vxencap` Specifies default values for the `format`, `privlen`, `privoffset` and `puboffset` parameters to the `vxencap` and `vxlvencap` commands. These commands are used when disks with existing partitions or slices are encapsulated, or when LVM disks are converted to VM disks. It is also called implicitly by the `vxdiskadm`, `vxconvert` (on AIX) and `vxvmconvert` (on HP-UX) commands, and by the SFM.

The following is a sample `vxencap` defaults file:

```
format=sliced
privlen=4096
privoffset=0
puboffset=1
```

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

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

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

In the defaults files, a line that is empty, or that begins with a “#” character in the first column, is treated as a comment, and is ignored.

Apart from comment lines, all other lines must define attributes and their values using the format `attribute=value`. Each line starts in the first column, and is terminated by the value. No white space is allowed around the = sign.

Maintaining your system

You may need to perform maintenance tasks on the CDS disks and CDS disk groups. Refer to the respective section for each type of task.

- Disk tasks

See “[Disk tasks](#)” on page 469.

- Disk group tasks
See “[Disk group tasks](#)” on page 471.
- Displaying information
See “[Displaying information](#)” on page 477.
- Default activation mode of shared disk groups
See “[Default activation mode of shared disk groups](#)” on page 480.
- Additional considerations when importing CDS disk groups
See “[Defaults files](#)” on page 467.

Disk tasks

The following disk tasks are supported:

- [Changing the default disk format](#)
- [Restoring CDS disk labels](#)

Changing the default disk format

When disks are put under VxVM control, they are formatted with the default `cdsdisk` layout. This happens during the following operations:

- Initialization of disks
- Encapsulation of disks with existing partitions or slices (Linux and Solaris systems)
- Conversion of LVM disks (AIX, HP-UX and Linux systems)

You can override this behavior by changing the settings in the system defaults files. For example, you can change the default format to `sliced` for disk initialization by modifying the definition of the `format` attribute in the `/etc/default/vxdisk` defaults file.

To change the default format for disk encapsulation or LVM disk conversion

- Edit the `/etc/default/vxencap` defaults file, and change the definition of the `format` attribute.
See “[Defaults files](#)” on page 467.

Restoring CDS disk labels

CDS disks have the following labels:

- Platform block

- AIX coexistence label
- HP-UX coexistence or VxVM ID block

There are also backup copies of each. If any of the primary labels become corrupted, VxVM will not bring the disk online and user intervention is required.

If two labels are intact, the disk is still recognized as a `cdsdisk` (though in the error state) and `vxdisk flush` can be used to restore the CDS disk labels from their backup copies.

Note: For disks larger than 1 TB, `cdsdisks` use the EFI layout. The procedure to restore disk labels does not apply to `cdsdisks` with EFI layout.

Primary labels are at sectors 0, 7, and 16; and a normal flush will not flush sectors 7 and 16. Also, the private area is not updated as the disk is not in a disk group. There is no means of finding a “good” private region to flush from. In this case, it is possible to restore the CDS disk labels from the existing backups on disk using the flush operation.

If a corruption happened after the labels were read and the disk is still online and part of a disk group, then a flush operation will also flush the private region.

Warning: Caution and knowledge must be employed because the damage could involve more than the CDS disk labels. If the damage is constrained to the first 128K, the disk flush would fix it. This could happen if another system on the fabric wrote a disk label to a disk that was actually a CDS disk in some disk group.

To rewrite the CDS ID information on a specific disk

- Type the following command:

```
# vxdisk flush disk_access_name
```

This rewrites all labels except sectors 7 and 16.

To rewrite all the disks in a CDS disk group

- Type the following command:

```
# vxdg flush diskgroup
```

This rewrites all labels except sectors 7 and 16.

To forcibly rewrite the AIX coexistence label in sector 7 and the HP-UX coexistence label or VxVM ID block in sector 16

- Type the following command:

```
# vxdisk -f flush disk_access_name
```

This command rewrites all labels if there exists a valid VxVM ID block that points to a valid private region. The `-f` option is required to rewrite sectors 7 and 16 when a disk is taken offline due to label corruption (possibly by a Windows system on the same fabric).

Disk group tasks

The following disk group tasks are supported:

- [Changing the alignment of a disk group during disk encapsulation](#)
- [Changing the alignment of a non-CDS disk group](#)
- [Determining the setting of the CDS attribute on a disk group](#)
- [Splitting a CDS disk group](#)
- [Moving objects between CDS disk groups and non-CDS disk groups](#)
- [Moving objects between CDS disk groups](#)
- [Joining disk groups](#)
- [Changing the default CDS setting for disk group creation](#)
- [Creating non-CDS disk groups](#)
- [Upgrading an older version non-CDS disk group](#)
- [Replacing a disk in a CDS disk group](#)
- [Setting the maximum number of devices for CDS disk groups](#)

Changing the alignment of a disk group during disk encapsulation

If you use the `vxdiskadm` command to encapsulate a disk into a disk group with an alignment of 8K, the disk group alignment must be reduced to 1.

If you use the `vxencap` command to perform the encapsulation, the alignment is carried out automatically without a confirmation prompt.

To change the alignment of a disk group during disk encapsulation

- Run the `vxdiskadm` command, and select the “Add or initialize one or more disks” item from the main menu. As part of the encapsulation process, you are asked to confirm that a reduction of the disk group alignment from 8K to 1 is acceptable.

Changing the alignment of a non-CDS disk group

The alignment value can only be changed for disk groups with version 110 or greater.

For a CDS disk group, `alignment` can only take a value of `8k`. Attempts to set the alignment of a CDS disk group to `1` fail unless you first change it to a non-CDS disk group.

Increasing the alignment may require `vxcdsconvert` to be run to change the layout of the objects in the disk group.

To display the current alignment value of a disk group, use the `vxprint` command.

See [“Displaying the disk group alignment”](#) on page 479.

To change the alignment value of a disk group

- Type the `vx dg set` command:

```
# vx dg -g diskgroup set align={1|8k}
```

The operation to increase the alignment to 8K fails if objects exist in the disk group that do not conform to the new alignment restrictions. In that case, use the `vxcdsconvert alignment` command to change the layout of the objects:

```
# vxcdsconvert -g diskgroup [-A] [-d defaults_file] \  
  [-o novolstop] alignment [attribute=value] ...
```

This command increases the alignment value of a disk group and its objects to 8K, without converting the disks.

The sequence 8K to 1 to 8K is possible only using `vx dg set` as long as the configuration does not change after the 8K to 1 transition.

See [“Converting a non-CDS disk group to a CDS disk group”](#) on page 464.

Splitting a CDS disk group

You can use the `vx dg split` command to create a CDS disk group from an existing CDS disk group. The new (target) disk group preserves the setting of the CDS attribute and alignment in the original (source) disk group.

To split a CDS disk group

- Use the `vx dg split` command to split CDS disk groups.
See the *Veritas Volume Manager Administrator’s Guide*.

Moving objects between CDS disk groups and non-CDS disk groups

The alignment of a source non-CDS disk group must be 8K to allow objects to be moved to a target CDS disk group. If objects are moved from a CDS disk group to a target non-CDS disk group with an alignment of 1, the alignment of the target disk group remains unchanged.

To move objects between a CDS disk group and a non-CDS disk group

- Use the `vxdg move` command to move objects between a CDS disk group and a non-CDS disk groups.

See the *Veritas Volume Manager Administrator's Guide*.

Moving objects between CDS disk groups

The disk group alignment does not change as a result of moving objects between CDS disk groups.

To move objects between CDS disk groups

- Use the `vxdg move` command to move objects between CDS disk groups.

See the *Veritas Volume Manager Administrator's Guide*.

Joining disk groups

Joining two CDS disk groups or joining two non-CDS disk groups is permitted, but you cannot join a CDS disk group to a non-CDS disk group. If two non-CDS disk groups have different alignment values, the alignment of the resulting joined disk group is set to 1, and an informational message is displayed.

To join two disk groups

- Use the `vxdg join` command to join two disk groups.

See the *Veritas Volume Manager Administrator's Guide*.

Changing the default CDS setting for disk group creation

To change the default CDS setting for disk group creation

- Edit the `/etc/default/vxdg` file, and change the setting for the `cds` attribute.

Creating non-CDS disk groups

A disk group with a version lower than 110 is given an alignment value equal to 1 when it is imported. This is because the `dg_align` value is not stored in the configuration database for such disk groups.

To create a non-CDS disk group with a version lower than 110

- Type the following `vxdg` command:

```
# vxdg -T version init diskgroup disk_name=disk_access_name
```

Upgrading an older version non-CDS disk group

You may want to upgrade a non-CDS disk group with a version lower than 110 in order to use new features other than CDS. After upgrading the disk group, the `cds` attribute is set to `off`, and the disk group has an alignment of 1.

Note: You must also perform a disk group conversion (using the `vxcdsconvert` utility) to use the CDS feature.

To upgrade the non-CDS pre-version 110 disk group

- Type the following `vxdg` command:

```
# vxdg upgrade diskgroup
```

Replacing a disk in a CDS disk group

Note: When replacing a disk in a CDS disk group, you cannot use a non-CDS disk as the replacement.

To replace a disk in a CDS disk group

- Type the following commands:

```
# vxdg -g diskgroup -k rmdisk disk_name  
# vxdg -g diskgroup -k adddisk disk_name=disk_access_name
```

The `-k` option retains and reuses the disk media record for the disk that is being replaced. The following example shows a disk device `disk21` being reassigned to disk `mydg01`.

```
# vxdg -g diskgroup -k rmdisk mydg01  
# vxdg -g diskgroup -k adddisk mydg01=disk21
```

For other operating systems, use the appropriate device name format.

Setting the maximum number of devices for CDS disk groups

To set the maximum number of devices that can be created in a CDS disk group

- Type the following `vxchg set` command:

```
# vxchg -g diskgroup set maxdev=max-devices
```

The `maxdev` attribute can take any positive integer value that is greater than the number of devices that are currently in the disk group.

Changing the DRL map and log size

If DRL is enabled on a newly-created volume without specifying a log or map size, default values are used. You can use the command line attributes `logmap_len` and `loglen` in conjunction with the `vxassist`, `vxvol`, and `vxmake` commands to set the DRL map and DRL log sizes. The attributes can be used independently, or they can be combined.

You can change the DRL map size and DRL log size only when the volume is disabled and the DRL maps are not in use. Changes can be made to the DRL map size only for volumes in a CDS disk group.

The `logmap_len` attribute specifies the required size, in bytes, for the DRL log. It cannot be greater than the number of bytes available in the map on the disk.

To change the DRL map and log size

- Use the following commands to remove and rebuild the logs:

```
# vxassist -g diskgroup remove log volume nlog=0
# vxassist -g diskgroup addlog volume nlog=nlogs \
  logtype=drl logmap_len=len-bytes [loglen=len-blocks]
```

Note the following restrictions

If only `logmap_len` is specified

The DRL log size is set to the default value (33 * disk group alignment).

If `logmap_len` is greater than (DRL log size) / 2

The command fails, and you need to either provide a sufficiently large `loglen` value or reduce `logmap_len`.

For CDS disk groups

The DRL map and log sizes are set to a minimum of 2 * (disk group alignment).

Creating a volume with a DRL log

To create a volume with a traditional DRL log by using the `vxassist` command

- Type the following command:

```
# vxassist -g diskgroup make volume length mirror=2 \  
logtype=drl [loglen=len-blocks] [logmap_len=len-bytes]
```

This command creates log subdisks that are each equal to the size of the DRL log.

Note the following restrictions

If neither `logmap_len` nor `loglen` is specified

- `loglen` is set to a default value that is based on disk group alignment.
- `maplen` is set to a reasonable value.

If only `loglen` is specified

- For pre-version 110 disk groups, `maplen` is set to zero.
- For version 110 and greater disk groups, `maplen` is set to use all the bytes available in the on-disk map.

If only `logmap_len` is specified

- For pre-version 110 disk groups, `logmap_len` is not applicable.
- For version 110 and greater disk groups, `maplen` must be less than the number of available bytes in the on-disk map for the default log length.

Setting the DRL map length

To set a DRL map length

- 1 Stop the volume to make the DRL inactive.
- 2 Type the following command:

```
# vxvol -g diskgroup set [loglen=len-blocks] \  
[logmap_len=len-bytes] volume
```

This command does not change the existing DRL map size.

Note the following restrictions

If both `logmap_len` and `loglen` are specified

- if `logmap_len` is greater than `loglen/2`, `vxvol` fails with an error message. Either increase `loglen` to a sufficiently large value, or decrease `logmap_len` to a sufficiently small value.
- The value of `logmap_len` cannot exceed the number of bytes in the on-disk map.

If `logmap_len` is specified

- The value is constrained by size of the log, and cannot exceed the size of the on-disk map. The size of the on-disk map in blocks can be calculated from the following formula:
$$\text{round}(\text{loglen}/\text{nmaps}) - 24$$
where `nmaps` is 2 for a private disk group, or 33 for a shared disk group.
- The value of `logmap_len` cannot exceed the number of bytes in the on-disk map.

If `loglen` is specified

- Specifying a value that is less than twice the disk group alignment value results in an error message.
- The value is constrained by size of the logging subdisk.

Displaying information

This section describes the following tasks:

- [Determining the setting of the CDS attribute on a disk group](#)
- [Displaying the maximum number of devices in a CDS disk group](#)
- [Displaying map length and map alignment of traditional DRL logs](#)
- [Displaying the disk group alignment](#)
- [Displaying the log map length and alignment](#)
- [Displaying offset and length information in units of 512 bytes](#)

Determining the setting of the CDS attribute on a disk group

To determine the setting of the CDS attribute on a disk group

- Use the `vxdg list` command or the `vxprint` command to determine the setting of the CDS attribute, as shown in the following examples:

```
# vxdg list

NAME                STATE                ID
dgTestSol2         enabled,cds         1063238039.206.vmescl

# vxdg list dgTestSol2

Group:      dgTestSol2
dgid:      1063238039.206.vmescl
import-id: 1024.205
flags:     cds
version:   110
alignment: 8192 (bytes)
.
.
.

# vxprint -F %cds -G -g dgTestSol2

on
```

The disk group, `dgTestSol2`, is shown as having the CDS flag set.

Displaying the maximum number of devices in a CDS disk group

To display the maximum number of devices in a CDS disk group

- Type the following command:

```
# vxprint -g diskgroup -G -F %maxdev
```

Displaying map length and map alignment of traditional DRL logs

To display the map length and map alignment of traditional DRL logs

- Type the following commands

```
# vxprint -g diskgroup -vl volume
# vxprint -g diskgroup -vF '%name %logmap_len %logmap_align' \
volume
```

Displaying the disk group alignment

To display the disk group alignment

- Type the following command:

```
# vxprint -g diskgroup -G -F %align
```

Utilities such as `vxprint` and `vx dg list` that print information about disk group records also output the disk group alignment.

Displaying the log map length and alignment

To display the log map length and alignment

- Type the following command:

```
# vxprint -g diskgroup -lv volume
```

For example, to print information for the volume `vol1` in disk group `dg1`:

```
# vxprint -g dg1 -lv vol1
```

The output is of the form:

```
logging: type=REGION loglen=0 serial=0/0 mapalign=0
maplen=0 (disabled)
```

This indicates a log map alignment (`logmap_align`) value of 0, and a log map length (`logmap_len`) value of 0.

If the log map is set and enabled, the command and results may be in the following form:

```
# vxprint -lv drlvol
```

```
Disk group: dgTestSol
Volume:    drlvol
info:      len=20480
type:      usetype=fsgen
state:     state=ACTIVE kernel=ENABLED cdsrecovery=0/0 (clean)
assoc:     plexes=drlvol-01,drlvol-02,drlvol-03
policies:  read=SELECT (round-robin) exceptions=GEN_DET_SPARSE
flags:     closed writecopy writeback
```

```
logging:  type=REGION loglen=528 serial=0/0 mapalign=16
maplen=512 (enabled)
apprecov: seqno=0/0
recovery: mode=default
recov_id=0
device:  minor=46000 bdev=212/46000 cdev=212/46000
path=/dev/vx/dsk/dgTestSol/drlvol
perms:   user=root group=root mode=0600
guid:    {d968de3e-1dd1-11b2-8fc1-080020d223e5}
```

Displaying offset and length information in units of 512 bytes

To display offset and length information in units of 512 bytes

- Specify the `-b` option to the `vxprint` and `vxdisk` commands, as shown in these examples:

```
# vxprint -bm
# vxdisk -b list
```

Specifying the `-b` option enables consistent output to be obtained on different platforms. Without the `-b` option, the information is output in units of sectors. The number of bytes per sector differs between platforms.

When the `vxprint -bm` or `vxdisk -b list` command is used, the output also contains the `b` suffix, so that the output can be fed back to `vxmake`.

Default activation mode of shared disk groups

The default activation mode of shared disk groups involves a local in-kernel policy that differs between platforms. This means that, regardless of the platform on which the disk group was created, the importing platform will have platform-specific behavior with respect to activation of shared disk groups. Specifically, with the exception of HP-UX, importing a shared disk group results in the volumes being active and enabled for shared-write. In the case of HP-UX, the shared volumes will be inactive and require other actions to activate them for shared-write operations.

Additional considerations when importing CDS disk groups

Before you attempt to use CDS to move disk groups between different operating systems, and if the configuration of the disks has changed since the target system was last rebooted, you should consider the following points

Does the target system know about the disks?

For example, the disks may not have been connected to the system either physically (not cabled) or logically (using FC zoning or LUN masking) when the system was booted up, but they have subsequently been connected without rebooting the system. This can happen when bringing new storage on-line, or when adding an additional DMP path to existing storage. On the target system, both the operating system and VxVM must be informed of the existence of the new storage. Issue the appropriate command to tell the operating system to look for the storage. (On Linux, depending on the supported capabilities of the host adapter, you may need to reboot the target system to achieve this.) Having done this, run either of the following commands on the target system to have VxVM recognize the storage:

```
# vxdctl enable
# vxdisk scandisks
```

Do the disks contain partitions or slices?

Both the Solaris and Linux operating systems maintain information about partitions or slices on disks. If you repartition a disk after the target system was booted, use the appropriate command to instruct the operating system to rescan the disk's TOC or partition table. For example, on a target Linux system, use the following command:

```
# blockdev --rereadpt
```

Having done this, run either of the following commands on the target system to have VxVM recognize the storage:

```
# vxdctl enable
# vxdisk scandisks
```

Has the format of any of the disks changed since the target system was last booted?

For example, if you use the `vxdisksetup -i` command to format a disk for VxVM on one system, the `vxdisk list` command on the target system may still show the format as being `auto:none`. If so, use either of the following commands on the target system to instruct VxVM to rescan the format of the disks:

```
# vxdctl enable
# vxdisk scandisks
```

File system considerations

To set up or migrate volumes with VxFS file systems with CDS, you must consider the file system requirements. This section describes these requirements. It also describes additional tasks required for migrating or setting up in CDS.

Considerations about data in the file system

Data within a file system might not be in the appropriate format to be accessed if moved between different types of systems. For example, files stored in proprietary binary formats often require conversion for use on the target platform. Files containing databases might not be in a standard format that allows their access when moving a file system between various systems, even if those systems use the same byte order. Oracle 10g's Cross-Platform Transportable Tablespace is a notable exception; if used, this feature provides a consistent format across many platforms.

Some data is inherently portable, such as plain ASCII files. Other data is designed to be portable and the applications that access such data are able to access it irrespective of the system on which it was created, such as Adobe PDF files.

Note that the CDS facilities do not convert the end user data. The data is uninterpreted by the file system. Only individual applications have knowledge of the data formats, and thus those applications and end users must deal with this issue. This issue is not CDS-specific, but is true whenever data is moved between different types of systems.

Even though a user might have a file system with data that cannot be readily interpreted or manipulated on a different type of system, there still are reasons for moving such data by using CDS mechanisms. For example, if the desire is to bring a file system off line from its primary use location for purposes of backing it up without placing that load on the server or because the system on which it will be backed up is the one that has the tape devices directly attached to it, then using CDS to move the file system is appropriate.

An example is a principal file server that has various file systems being served by it over the network. If a second file server system with a different operating system was purchased to reduce the load on the original server, CDS can migrate the file system instead of having to move the data to different physical storage over the network, even if the data could not be interpreted or used by either the original or new file server. This is a scenario that often occurs when the data is only accessible or understood by software running on PCs and the file server is UNIX or Linux-based.

File system migration

File system migration refers to the system management operations related to stopping access to a file system, and then restarting these operations to access the file system from a different computer system. File system migration might be required to be done once, such as when permanently migrating a file system to another system without any future desire to move the file system back to its original system or to other systems. This type of file system migration is referred

to as one-time file system migration. When ongoing file system migration between multiple systems is desired, this is known as ongoing file system migration. Different actions are required depending on the kind of migration, as described in the following sections.

Specifying the migration target

Most of the operations performed by the CDS commands require the target to which the file system is to be migrated to be specified by target specifiers in the following format:

```
os_name=name[,os_rel=release][,arch=arch_name]
[,vxfs_version=version][,bits=nbits]
```

The CDS commands require the following target specifiers:

<code>os_name=name</code>	Specifies the name of the target operating system to which the file system is planned to be migrated. Possible values are HP-UX, AIX, SunOS, or Linux. The <code>os_name</code> field must be specified if the target is specified.
<code>os_rel=release</code>	Specifies the operating system release version of the target. For example, 11.31.
<code>arch=arch_name</code>	Specifies the architecture of the target. For example, specify <code>ia</code> or <code>pa</code> for HP-UX.
<code>vxfs_version=version</code>	Specifies the VxFS release version that is in use at the target. For example, 5.1.
<code>bits=nbits</code>	Specifies the kernel bits of the target. <code>nbits</code> can have a value of 32 or 64 to indicate whether the target is running a 32-bit kernel or 64-bit kernel.

While `os_name` must be specified for all `fscdsadm` invocations that permit the target to be specified, all other target specifiers are optional and are available for the user to fine tune the migration target specification.

The CDS commands use the limits information available in the default CDS limits file, `/etc/vx/cdslimitstab`. If the values for the optional target specifiers are not specified, `fscdsadm` will choose the defaults for the specified target based on the information available in the limits file that best fits the specified target, and proceed with the CDS operation. The chosen defaults are displayed to the user before proceeding with the migration.

Note: The default CDS limits information file, `/etc/vx/cdslimitstab`, is installed as part of the VxFS package. The contents of this file are used by the VxFS CDS commands and should not be altered.

Examples of target specifications

The following are examples of target specifications:

<code>os_name=AIX</code>	Specifies the target operating system and uses defaults for the remainder.
<code>os_name=HP-UX, os_rel=11.23, arch=ia, vxfs_version=5.0, bits=64</code>	Specifies the operating system, operating system release version, architecture, VxFS version, and kernel bits of the target.
<code>os_name=SunOS, arch=sparc</code>	Specifies the operating system and architecture of the target.
<code>os_name=Linux, bits=32</code>	Specifies the operating system and kernel bits of the target.

Using the `fscdsadm` command

The `fscdsadm` command can be used to perform the following CDS tasks:

- [Checking that the metadata limits are not exceeded](#)
- [Maintaining the list of target operating systems](#)
- [Enforcing the established CDS limits on a file system](#)
- [Ignoring the established CDS limits on a file system](#)
- [Validating the operating system targets for a file system](#)
- [Displaying the CDS status of a file system](#)

Checking that the metadata limits are not exceeded

To check that the metadata limits are not exceeded

- Type the following command to check whether there are any file system entities with metadata that exceed the limits for the specified target operating system:

```
# fscdsadm -v -t target mount_point
```

Maintaining the list of target operating systems

When a file system is migrated on an ongoing basis between multiple systems, the types of operating systems that are involved in these migrations are maintained in a `target_list` file. Knowing what these targets are allows VxFS to determine file system limits that are appropriate to all of these targets. The file system limits that are enforced are file size, user ID, and group ID. The contents of the `target_list` file are manipulated by using the `fscdsadm` command.

Adding an entry to the list of target operating systems

To add an entry to the list of target operating systems

- Type the following command:

```
# fscdsadm -o add -t target mount_point
```

See [“Specifying the migration target”](#) on page 483.

Removing an entry from the list of target operating systems

To remove an entry from the list of target operating systems

- Type the following command:

```
# fscdsadm -o remove -t target mount_point
```

See [“Specifying the migration target”](#) on page 483.

Removing all entries from the list of target operating systems

To remove all entries from the list of target operating systems

- Type the following command:

```
# fscdsadm -o none mount_point
```

Displaying the list of target operating systems

To display a list of all target operating systems

- Type the following command:

```
# fscdsadm -o list mount_point
```

Enforcing the established CDS limits on a file system

By default, CDS ignores the limits that are implied by the operating system targets that are listed in the `target_list` file.

To enforce the established CDS limits on a file system

- Type the following command:

```
# fscdsadm -l enforce mount_point
```

Ignoring the established CDS limits on a file system

By default, CDS ignores the limits that are implied by the operating system targets that are listed in the `target_list` file.

To ignore the established CDS limits on a file system

- Type the following command:

```
# fscdsadm -l ignore mount_point
```

Validating the operating system targets for a file system

To validate the operating system targets for a file system

- Type the following command:

```
# fscdsadm -v mount_point
```

Displaying the CDS status of a file system

The CDS status that is maintained for a file system includes the following information:

- the `target_list` file
- the limits implied by the `target_list` file
- whether the limits are being enforced or ignored
- whether all files are within the CDS limits for all operating system targets that are listed in the `target_list` file

To display the CDS status of a file system

- Type the following command:

```
# fscdsadm -s mount_point
```

Migrating a file system one time

This example describes a one-time migration of data between two operating systems. Some of the following steps require a backup of the file system to be created. To simplify the process, you can create one backup before performing any of the steps instead of creating multiple backups as you go.

To perform a one-time migration

- 1 If the underlying Volume Manager storage is not contained in a CDS disk group, it must first be upgraded to be a CDS disk group, and all other physical considerations related to migrating the storage physically between systems must first be addressed.

See [“Converting a non-CDS disk group to a CDS disk group”](#) on page 464.

- 2 If the file system is using a disk layout version prior to 7, upgrade the file system to Version 7.

See the *Veritas Storage Foundation Installation Guide*.

- 3 Use the following command to ensure that there are no files in the file system that will be inaccessible after migrating the data due to large file size or differences in user or group ID between platforms:

```
# fscdsadm -v -t target mount_point
```

If such files exist, move the files to another file system or reduce the size of the files.

- 4 Unmount the file system:

```
# umount mount_point
```

- 5 Use the `fscdsconv` command to convert the file system to the opposite endian.

See [“Converting the byte order of a file system”](#) on page 489.

- 6 Make the physical storage and Volume Manager logical storage accessible on the Linux system by exporting the disk group from the source system and importing the disk group on the target system after resolving any other physical storage attachment issues.

See [“Disk tasks”](#) on page 469.

- 7 Mount the file system on the target system.

Migrating a file system on an ongoing basis

This example describes how to migrate a file system between platforms on an ongoing basis. Some of the following steps require a backup of the file system to

be created. To simplify the process, you can create one backup before performing any of the steps instead of creating multiple backups as you go.

To perform an ongoing migration

- 1 Use the following command to ensure that there are no files in the file system that will be inaccessible after migrating the data due to large file size or differences in user or group ID between platforms:

```
# fscdsadm -v -t target mount_point
```

If such files exist, move the files to another file system or reduce the size of the files.

- 2 Add the platform on the `target_list` file:

- If migrating a file system between Solaris and Linux, add `SunOS` and `Linux` to the `target_list` file:

```
# fscdsadm -o add -t os_name=SunOS /mnt1  
# fscdsadm -o add -t os_name=Linux /mnt1
```

- If migrating a file system between HP-UX and Linux, add `HP-UX` and `Linux` to the `target_list` file:

```
# fscdsadm -o add -t os_name=HP-UX /mnt1  
# fscdsadm -o add -t os_name=Linux /mnt1
```

- 3 Enforce the limits:

```
# fscdsadm -l enforce mount_point
```

This is the last of the preparation steps. When the file system is to be migrated, it must be unmounted, and then the storage moved and mounted on the target system.

- 4 Unmount the file system:

```
# umount mount_point
```

- 5 Make the file system suitable for use on the specified target.

See [“Converting the byte order of a file system”](#) on page 489.

- 6 Make the physical storage and Volume Manager logical storage accessible on the target system by exporting the disk group from the source system and importing the disk group on the target system after resolving any other physical storage attachment issues.
See “Disk tasks” on page 469.
- 7 Mount the file system on the target system.

Stopping ongoing migration

To stop performing ongoing migration

- ◆ Type the following commands:

```
# fscdsadm -l ignore mount_point
# fscdsadm -o none mount_point
```

The file system is left on the current system.

When to convert a file system

When moving a file system between two systems, it is essential to run the `fscdsconv` command to perform all of the file system migration tasks. The `fscdsconv` command validates the file system to ensure that it does not exceed any of the established CDS limits on the target, and converts the byte order of the file system if the byte order of the target is opposite to that of the current system.

Warning: Prior to VxFS 4.0 and disk layout Version 6, VxFS did not officially support moving file systems between different platforms, although in many cases a user may have successfully done so. Do not move file systems between platforms when using versions of VxFS prior to Version 4, or when using disk layouts earlier than Version 6. Instead, upgrade to VxFS 4.0 or higher, and disk layout Version 6 or later. Failure to upgrade before performing cross-platform movement can result in data loss or data corruption.

Note: If you replicate data from a little-endian to a big-endian system (or vice versa), you must convert the application after the replication completes.

Converting the byte order of a file system

Use the `fscdsconv` command to migrate a file system from one system to another.

To convert the byte order of a file system

- 1 Determine the disk layout version of the file system that you will migrate:

```
# fstyp -v /dev/vx/rdisk/diskgroup/volume | grep version  
  
magic a501fcf5 version 7 ctime Thu Jun 1 16:16:53 2006
```

Only file systems with Version 6 or later disk layout can be converted. If the file system has an earlier disk layout version, convert the file system to Version 6 or Version 7 disk layout before proceeding.

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

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

- 2 Perform a full file system back up. Failure to do so could result in data loss or data corruption under some failure scenarios in which restoring from the backup is required.
- 3 Designate a file system with free space where `fscdsconv` may create a file that will contain recovery information for usage in the event of a failed conversion.

Depending on the nature of the file system to be converted, for example if it is mirrored, you may wish to designate the recovery file to reside in a file system with the same level of failure tolerance. Having the same level of failure tolerance reduces the number of failure scenarios that would require restoration from the backup.

- 4 Unmount the file system to be converted:

```
# umount mount_point
```

- 5 Use the `fscdsconv` command to export the file system to the required target:

```
# fscdsconv -f recovery_file -t target -e special_device
```

`target` specifies the system to which you are migrating the file system.

See [“Specifying the migration target”](#) on page 483.

`recovery_file` is the name of the recovery file to be created by the `fscdsconv` command. `special_device` is the raw device or volume that contains the file system to be converted.

Include the file system that you chose in 3 when designating the recovery file.

For example, if the file system chosen to contain the recovery file is mounted on `/data/fs3`, the recovery file could be specified as

`/data/fs3/jan04recovery`. If there is not enough disk space on the chosen file system for the recovery file to be created, the conversion aborts and the file system to be converted is left intact.

The recovery file is not only used for recovery purposes after a failure, but is also used to perform the conversion. The directory that will contain the recovery file should not allow non-system administrator users to remove or replace the file, as this could lead to data loss or security breaches. The file should be located in a directory that is not subject to system or local scripts will remove the file after a system reboot, such as that which occurs with the `/tmp` and `/var/tmp` directories on the Solaris operating system.

The recovery file is almost always a sparse file. The disk utilization of this file can best be determined by using the following command:

```
# du -sk filename
```

The recovery file is used only when the byte order of the file system must be converted to suit the specified migration target.

- 6 If you are converting multiple file systems at the same time, which requires the use of one recovery file per file system, record the names of the recovery files and their corresponding file systems being converted in the event that recovery from failures is required at a later time.
- 7 Based on the information provided regarding the migration target, `fscdsconv` constructs and displays the complete migration target and prompts the user to verify all details of the target. If the migration target must be changed, enter `n` to exit `fscdsconv` without modifying the file system. At this point in the process, `fscdsconv` has not used the specified recovery file.

- 8 If the byte order of the file system must be converted to migrate the file system to the specified target, `fscdsconv` prompts you to confirm the migration. Enter `y` to convert the byte order of the file system. If the byte order does not need to be converted, a message displays indicating this fact.
- 9 The `fscdsconv` command indicates if any files are violating the maximum file size, maximum UID, or maximum GID limits on the specified target and prompts you if it should continue. If you must take corrective action to ensure that no files violate the limits on the migration target, enter `n` to exit `fscdsconv`. At this point in the process, `fscdsconv` has not used the specified recovery file.

If the migration converted the byte order of the file system, `fscdsconv` created a recovery file. The recovery file is not removed after the migration completes, and can be used to restore the file system to its original state if required at a later time.

- 10 If a failure occurs during the conversion, the failure could be one of the following cases:

- System failure.
- `fscdsconv` failure due to program defect or abnormal termination resulting from user actions.

In such cases, the file system being converted is no longer in a state in which it can be mounted or accessed by normal means through other VxFS utilities. To recover the file system, invoke the `fscdsconv` command with the recovery flag, `-r`:

```
# fscdsconv -r -f recovery_file special_device
```

When the `-r` flag is specified, `fscdsconv` expects the recovery file to exist and that the file system being converted is the same file system specified in this second invocation of `fscdsconv`.

- 11 After invoking `fscdsconv` with the `-r` flag, the conversion process will restart and complete, given no subsequent failures.

In the event of another failure, repeat 10.

Under some circumstances, you will be required to restore the file system from the backup, such as if the disk fails that contains the recovery file. Failure to have created a backup would then result in total data loss in the file system. I/O errors on the device that holds the file system would also require a backup to be restored after the physical device problems are addressed. There may be other causes of failure that would require the use of the backup.

Importing and mounting a file system from another system

The `fscdsconv` command can be used to import and mount a file system that was previously used on another system.

To import and mount a file system from another system

- ◆ Convert the file system:

```
# fscdsconv -f recovery_file -i special_device
```

If the byte order of the file system needs to be converted	Enter <code>y</code> to convert the byte order of the file system when prompted by <code>fscdsconv</code> . If the migration converted the byte order of the file system, <code>fscdsconv</code> creates a recovery file that persists after the migration completes. If required, you can use this file to restore the file system to its original state at a later time.
--	--

If the byte order of the file system does not need to be converted	A message displays that the byte order of the file system does not need to be converted.
--	--

Alignment value and block size

On the AIX, Linux and Solaris operating systems, an alignment value of 1 is equivalent to a block size of 512 bytes. On the HP-UX operating system, it is equivalent to a block size of 1024 bytes.

The block size on HP-UX is different from that on other supported platforms. Output from commands such as `vxdisk` and `vxprint` looks different on HP-UX for the same disk group if the `-b` option is not specified.

Migrating a snapshot volume

This example demonstrates how to migrate a snapshot volume containing a VxFS file system from a Solaris SPARC system (big endian) to a Linux system (little endian) or HP-UX system (big endian) to a Linux system (little endian).

To migrate a snapshot volume

- 1 Create the instant snapshot volume, `snapvol`, from an existing plex in the volume, `vol`, in the CDS disk group, `datadg`:

```
# vxsnap -g datadg make source=vol/newvol=snapvol/nmirror=1
```

- 2 Quiesce any applications that are accessing the volume. For example, suspend updates to the volume that contains the database tables. The database may have a hot backup mode that allows you to do this by temporarily suspending writes to its tables.
- 3 Refresh the plexes of the snapshot volume using the following command:

```
# vxsnap -g datadg refresh snapvol source=yes syncing=yes
```

- 4 The applications can now be unquiesced.

If you temporarily suspended updates to the volume by a database in 2, release all the tables from hot backup mode.

- 5 Use the `vxsnap syncwait` command to wait for the synchronization to complete:

```
# vxsnap -g datadg syncwait snapvol
```

- 6 Check the integrity of the file system, and then mount it on a suitable mount point:

```
# fsck -F vxfs /dev/vx/rdisk/datadg/snapvol  
# mount -F vxfs /dev/vx/dsk/datadg/snapvol /mnt
```

- 7 Confirm whether the file system can be converted to the target operating system:

```
# fscdstask validate Linux /mnt
```

- 8 Unmount the snapshot:

```
# umount /mnt
```

- 9 Convert the file system to the opposite endian:

```
# fscdsconv -f /tmp/fs_recov/recov.file /dev/vx/dsk/datadg/snapvol
```

This step is only required if the source and target systems have the opposite endian configuration.

- 10** Split the snapshot volume into a new disk group, `migdg`, and deport that disk group:

```
# vxdg split datadg migdg snapvol  
# vxdg deport migdg
```

- 11** Import the disk group, `migdg`, on the Linux system:

```
# vxdg import migdg
```

It may be necessary to reboot the Linux system so that it can detect the disks.

- 12** Use the following commands to recover and restart the snapshot volume:

```
# vxrecover -g migdg -m snapvol
```

- 13** Check the integrity of the file system, and then mount it on a suitable mount point:

```
# fsck -t vxfs /dev/vx/dsk/migdg/snapvol  
# mount -t vxfs /dev/vx/dsk/migdg/snapvol /mnt
```


Reference

- [Appendix A. Recovering from CDS errors](#)
- [Appendix B. Conversion error messages](#)
- [Appendix C. Files and scripts for sample scenarios](#)
- [Appendix D. Preparing a replica Oracle database](#)

Recovering from CDS errors

This appendix includes the following topics:

- [CDS error codes and recovery actions](#)

CDS error codes and recovery actions

[Table A-1](#) lists the CDS error codes and the action that is required.

Table A-1 Error codes and required actions

Error number	Message	Action
329	Cannot join a non-CDS disk group and a CDS disk group	Change the non-CDS disk group into a CDS disk group (or vice versa), then retry the join operation.
330	Disk group is for a different platform	Import the disk group on the correct platform. It cannot be imported on this platform.
331	Volume has a log which is not CDS compatible	To get a log which is CDS compatible, you need to stop the volume, if currently active, then start the volume. After the volume has been successfully started, retry setting the CDS attribute for the disk group.
332	License has expired, or is not available for CDS	Obtain a license from Symantec that enables the usage of CDS disk groups.

Table A-1 Error codes and required actions (*continued*)

Error number	Message	Action
333	Non-CDS disk cannot be placed in a CDS disk group	Do one of the following: <ul style="list-style-type: none"> ■ Add the disk to another disk group that is a non-CDS disk group. ■ Re-initialize the disk as a CDS disk so that it can be added to the CDS disk group. ■ Change the CDS disk group into a non-CDS disk group and then add the disk.
334	Disk group alignment not CDS compatible	Change the alignment of the disk group to 8K and then retry setting the CDS attribute for the disk group.
335	Subdisk length violates disk group alignment	Ensure that sub-disk length value is a multiple of 8K.
336	Subdisk offset violates disk group alignment	Ensure that sub-disk offset value is a multiple of 8K.
337	Subdisk plex offset violates disk group alignment	Ensure that sub-disk plex offset value is a multiple of 8K.
338	Plex stripe width violates disk group alignment	Ensure that plex stripe width value is a multiple of 8K.
339	Volume or log length violates disk group alignment	Ensure that the length of the volume is a multiple of 8K. For a log, set the value of the <code>dgalgn_checking</code> attribute to <code>round</code> . This ensures that the length of the log is silently rounded to a valid value.
340	Last disk media offset violates disk group alignment	Reassociate the DM record prior to upgrading.

Table A-1 Error codes and required actions (*continued*)

Error number	Message	Action
341	Too many device nodes in disk group	Increase the number of device nodes allowed in the disk group, if not already at the maximum. Otherwise, you need to remove volumes from the disk group, possibly by splitting the disk group.
342	Map length too large for current log length	Use a smaller map length for the DRL/DCM log, or increase the log length and retry.
343	Volume log map alignment violates disk group alignment	Remove the DRL/DCM log, then add it back after changing the alignment of the disk group.
345	Disk group contains an old-style RVG which cannot be imported on this platform	Import the disk group on the platform that created the RVG. To import the disk group on this platform, first remove the RVG on the creating platform.
346	Cache object autogrow by max_autogrow violates disk group alignment	Ensure that cache attribute value is a multiple of 8K.
347	User transactions are disabled for the disk group	Retry the command as it was temporarily disallowed by the <code>vxcdsconvert</code> command executing at the same time.
348	Disk is in use	Contact Technical Support.

Conversion error messages

This appendix includes the following topics:

- [List of conversion error messages](#)

List of conversion error messages

This appendix lists the error messages that you may encounter when converting LVM volume groups to VxVM disk groups and volumes. For each error message, a description is provided of the problem, and the action that you can take to troubleshoot it.

[Table B-1](#) shows the error messages that you may encounter during conversion.

Table B-1 Conversion error messages

Message	Description
<code>Analysis indicates that this volume group cannot be converted because not all of the disks and/or volumes in the LVM volume group are currently accessible</code>	<p>For successful conversion, all physical volumes in a volume group must be on-line, and all logical volumes must be active and accessible.</p> <p>Make sure the physical volumes in a volume group are on-line and the logical volumes are active and not in use.</p>

Table B-1 Conversion error messages (*continued*)

Message	Description
<p>Analysis shows that there is insufficient private space available to convert this volume group</p>	<p>The error message indicates the maximum amount of records that can be stored in the private space, and how many records are needed to convert this particular volume group.</p> <p>You can reduce the number of records needed by reducing the number of logical volumes in volume group by combining some of the logical volumes together.</p>
<p>The conversion process was unable to deactivate the volume group <i>vol_grp_name</i></p>	<p>This indicates that the conversion process cannot deactivate the volume group.</p> <p>The conversion cannot be completed without rebooting the machine. If you cannot afford to reboot, then choose abort and try again later.</p>
<p>This Volume Group contains one or more logical volumes with mirrored data</p>	<p>If you attempt to convert a Mirrored LVM Volume Group without a valid VxVM license installed, the conversion is not allowed.</p> <p>Install the required license before attempting the conversion.</p>
<p>Too many LVM Volumes to convert in this LVM Volume Group</p>	<p>If there is insufficient private space, the conversion is not allowed to continue. Also, the conversion records already generated are removed such that in the event of an unexpected crash and reboot, the conversion cannot proceed automatically.</p> <p>You can reduce the number of logical volumes in volume group by combining some of the logical volumes together, or by aborting. You can restart the conversion process later with fewer volumes in the group.</p>

Table B-1 Conversion error messages (*continued*)

Message	Description
<pre>vgchange: Couldn't deactivate volume group /dev/vol_grp</pre>	<p>The conversion process was unable to deactivate the volume group. The conversion cannot proceed without reboots being done. If you choose to not reboot your system, the conversion is aborted.</p> <p>The system responds with an option to complete the conversion by rebooting the system.</p>
<pre>vxdiskadm or vxconvert is already being run and these programs cannot run concurrently</pre>	<p>The system detects that the <code>vxdiskadd</code> program or the <code>vxconvert</code> program is already running.</p> <p>Retry at a later time. Otherwise, if you are certain that no other users are running either of these programs, remove the file <code>.DISKADD.LOCK</code> from the <code>/var/spool/locks</code> directory to allow you to run <code>vxconvert</code>.</p>

Files and scripts for sample scenarios

This appendix includes the following topics:

- [About files and scripts for sample scenarios](#)
- [Script to initiate online off-host backup of an Oracle database](#)
- [Script to put an Oracle database into hot backup mode](#)
- [Script to quiesce a Sybase ASE database](#)
- [Script to suspend I/O for a DB2 database](#)
- [Script to end Oracle database hot backup mode](#)
- [Script to release a Sybase ASE database from quiesce mode](#)
- [Script to resume I/O for a DB2 database](#)
- [Script to perform off-host backup](#)
- [Script to create an off-host replica Oracle database](#)
- [Script to complete, recover and start a replica Oracle database](#)
- [Script to start a replica Sybase ASE database](#)

About files and scripts for sample scenarios

This appendix contains the configuration files and scripts for the sample point-in-time copy processing scenarios described in this guide.

Note: These scripts are not supported by Symantec, and are provided for informational use only. You can purchase customization of the environment through Veritas Vpro Consulting Services.

Table C-1 list the files and scripts.

Table C-1 Files and scripts for sample scenarios

File or script	Used for...
<p>Sample script to initiate online offhost backup of an Oracle database.</p> <p>See “Script to initiate online off-host backup of an Oracle database” on page 509.</p>	<ul style="list-style-type: none"> ■ Online off-host backup. See “Making an off-host backup of an online database” on page 160.
<p>Sample scripts for Oracle, Sybase, or DB2.</p> <p>See “Script to put an Oracle database into hot backup mode” on page 511.</p> <p>See “Script to quiesce a Sybase ASE database” on page 512.</p> <p>See “Script to suspend I/O for a DB2 database” on page 512.</p>	<ul style="list-style-type: none"> ■ Online backup. See “About online database backup” on page 155. ■ Decision support. See “About decision support” on page 177.
<p>Sample scripts for Oracle, Sybase, or DB2.</p> <p>See “Script to end Oracle database hot backup mode” on page 513.</p> <p>See “Script to release a Sybase ASE database from quiesce mode” on page 513.</p> <p>See “Script to resume I/O for a DB2 database” on page 514.</p>	<ul style="list-style-type: none"> ■ Online backup. See “About online database backup” on page 155. ■ Decision support. See “About decision support” on page 177.
<p>Sample script to perform off-host backup.</p> <p>See “Script to perform off-host backup” on page 514.</p>	<ul style="list-style-type: none"> ■ Online off-host backup. See “Making an off-host backup of an online database” on page 160.

Table C-1 Files and scripts for sample scenarios (*continued*)

File or script	Used for...
<p>Sample script to create off-host replica Oracle database.</p> <p>See “Script to create an off-host replica Oracle database” on page 515.</p>	<ul style="list-style-type: none"> ■ Decision support. <p>See “Creating an off-host replica database” on page 184.</p>
<p>Sample scripts to start replica Oracle database or replica Sybase database.</p> <p>See “Script to complete, recover and start a replica Oracle database” on page 517.</p> <p>See “Script to start a replica Sybase ASE database” on page 519.</p>	<ul style="list-style-type: none"> ■ Decision support. <p>See “Creating an off-host replica database” on page 184.</p>

Script to initiate online off-host backup of an Oracle database

Use this script to initiate online off-host backup of an Oracle database.

```
#!/bin/ksh
#
# script: backup_online.sh <dbnode>
#
# Sample script for online, off-host backup.
#
# Note: This is not a production level script, its intention is to help
# you understand the procedure and commands for implementing
# an off-host point-in-time copy solution.

export ORACLE_SID=dbase
export ORACLE_HOME=/oracle/816
export PATH=$ORACLE_HOME/bin:$PATH

dbnode=$1
dbasedg=dbasedg
snapvoldg=snapdbdg
```

```
newvollist="snap_dbase_vol source=dbase_vol/newvol=snap_dbase_vol"
snapvollist="snap_dbase_vol"
volsnaplist="snap_dbase_vol source=dbase_vol"
exit_cnt=0
arch_loc=/archlog

# Put the Oracle database in hot-backup mode;
# see the backup_start.sh script for information.

su oracle -c backup_start.sh

# Refresh the snapshots of the volumes.
#
# Note: If the volume is not mounted, you can safely ignore the
# following message that is output by the snapshot operation:
#
# ERROR: Volume dbase_vol: No entry in /etc/mnttab for volume

vxsnap -g $dbasedg make $newvollist

# Take the database out of hot-backup mode;
# see the backup_end.sh script for information.

su oracle -c backup_end.sh

# Back up the archive logs that were generated while the database
# was in hot backup mode (as reported by the Oracle Server Manager).

# Move the snapshot volumes into a separate disk group.

vxvg split $dbasedg $snapdg $snapvollist

# Deport the snapshot disk group.

vxvg deport $snapdg

# The snapshots of the database can be imported and backed up
# on the OHP node and then deported.
# Note: Replace "rsh" with "remsh" on HP-UX systems.

rsh $dbnode -c "do_backup.sh $snapvollist"

# Import the snapshot disk group -- if the database disk group is
```

```
# cluster-shareable, you must also specify the -s option.

vxdg import $snapdg

# Join the snapshot disk group to the original volume disk group.

vxdg join $snapdg $dbasedg

# Restart the snapshot volumes.

for i in `echo $snapvollist`
do
    vxrecover -g $dbasedg -m $i
    vxvol -g $dbasedg start $i
done

# Reattach the snapshot volumes ready for the next backup cycle.

vxsnap -g $dbasedg reattach $volnaplist
```

Script to put an Oracle database into hot backup mode

Use this script to put an Oracle database into hot backup mode.

```
#!/bin/ksh
#
# script: backup_start.sh
#
# Sample script to put example Oracle database into hot backup mode.

export ORACLE_SID=dbase
export ORACLE_HOME=/oracle/816
export PATH=$ORACLE_HOME/bin:$PATH

svrmgrl <<!
connect internal
archive log list;
alter tablespace ts1 begin backup;

# .
# . Put all required tablespaces into hot backup mode
# .
```

```
alter tablespace tsN begin backup;  
quit  
!
```

Script to quiesce a Sybase ASE database

Use this script to quiesce a Sybase ASE database.

```
#!/bin/ksh  
#  
# script: backup_start.sh  
#  
# Sample script to quiesce example Sybase ASE database.  
#  
# Note: The "for external dump" clause was introduced in Sybase  
# ASE 12.5 to allow a snapshot database to be rolled forward.  
# See the Sybase ASE 12.5 documentation for more information.  
  
isql -Usa -Ppassword -SFMR <<!  
quiesce database tag hold database1[, database2]... [for external dump]  
go  
quit  
!
```

Script to suspend I/O for a DB2 database

Use this script to suspend I/O for a DB2 database.

```
#!/bin/ksh  
#  
# script: backup_start.sh  
#  
# Sample script to suspend I/O for a DB2 database.  
#  
# Note: To recover a database using backups of snapshots, the database  
# must be in LOGRETAIN mode.  
  
db2 <<!  
connect to database  
set write suspend for database  
quit  
!
```

Script to end Oracle database hot backup mode

Use this script to end Oracle database hot backup mode.

```
#!/bin/ksh
#
# script: backup_end.sh
#
# Sample script to end hot backup mode for example Oracle database.

export ORACLE_SID=dbase
export ORACLE_HOME=/oracle/816
export PATH=$ORACLE_HOME/bin:$PATH

svrmgrl <<!
connect internal
alter tablespace ts1 end backup;
# .
# . End hot backup mode for all required tablespaces.
# .
alter tablespace tsN end backup;
alter system switch logfile;
alter system switch logfile;
archive log list;
quit
!

# Note: The repeated line alter system switch logfile, forces
# a checkpoint and archives the contents of the redo logs
# recorded during the backup.
```

Script to release a Sybase ASE database from quiesce mode

Use this script to release a Sybase ASE database from quiesce mode.

```
#!/bin/ksh
#
# script: backup_end.sh
#
# Sample script to release example Sybase ASE database from quiesce mode.
```

```
isql -Usa -Ppassword -SEMR <<!  
quiesce database tag release  
go  
quit  
!
```

Script to resume I/O for a DB2 database

Use this script to resume I/O for a DB2 database.

```
#!/bin/ksh  
#  
# script: backup_end.sh  
#  
# Sample script to resume I/O for a DB2 database.  
#  
  
db2 <<!  
connect to database  
set write resume for database  
quit  
!
```

Script to perform off-host backup

Use this script to perform off-host backup.

```
#!/bin/ksh  
#  
# script: do_backup.sh <list_of_database_volumes>  
#  
# Sample script for off-host backup  
#  
# Note: This is not a production level script, its intention is to help  
# you understand the procedure and commands for implementing  
# an off-host point-in-time copy solution.  
  
# Modify the following procedure according to your environment  
# and backup method.  
  
snapvoldg=snapdbdg  
  
# Import the snapshot volume disk group.
```

```
vxvg import $snapvoldg

# Mount the snapshot volumes (the mount points must already exist).

for i in $*
do
    fsck -F vxfs /dev/vx/rdisk/$dbasedg/snap_$(i)
    mount -F vxfs /dev/vx/dsk/$dbasedg/snap_$(i) /bak/$(i)
done

# Back up each tablespace.
# back up /bak/ts1 &
...
# back up /bak/tsN &

wait

# Unmount snapshot volumes.

for i in $(echo $vollist)
do
    umount /bak/$(i)
done

# Deport snapshot volume disk group.

vxvg deport $snapvoldg

echo "do_backup over"
echo "\007 \007 \007 \007 \007 \007"
```

Script to create an off-host replica Oracle database

Use this script to create an off-host replica Oracle database.

```
#!/bin/ksh
#
# script: create_dss.sh <dbnode>
#
# Sample script to create a replica Oracle database on an OHP host.
#
# Note: This is not a production level script, its intention is to help
```

```
# you understand the procedure and commands for implementing
# an off-host point-in-time copy solution.

export ORACLE_SID=dbase
export ORACLE_HOME=/oracle/816
export PATH=$ORACLE_HOME/bin:$PATH

dbnode=$1
localdg=localdg
dbasedg=dbasedg
snapvoldg=snapdbdg
vollist="dbase_vol"
snapvollist="snap_dbase_vol"
volsnaplist="snap_dbase_vol source=dbase_vol"
exit_cnt=0
arch_loc=/archlog
rep_mnt_point=/rep

# Put the Oracle database in hot-backup mode;
# see the backup_start.sh script for information.

su oracle -c backup_start.sh

# Refresh the snapshots of the volumes.
#
# Note: If the volume is not mounted, you can safely ignore the
# following message that is output by the snapshot operation:
#
# vxvm:vxsync: ERROR: Volume dbase_vol: No entry in /etc/mnttab for volume

vxsnap -g $dbasedg refresh $volsnaplist

# Take the Oracle database out of hot-backup mode;
# see the backup_end.sh script for information.

su oracle -c backup_end.sh

# Move the snapshot volumes into a separate disk group.

vxvg split $dbasedg $snapdg $vollist

# Deport the snapshot disk group.
```

```
vxdg deport $snapdg

# Copy the archive logs that were generated while the database was
# in hot backup mode (as reported by the Oracle Server Manager) to the
# archive log location for the replica database on the OHP node
# (in this example, /rep/archlog).

rcp ${arch_loc}/* $dbnode:${rep_mnt_point}${arch_loc}

# The snapshots of the database can be now imported on the OHP node
# and used to complete, recover and start the replica database.
# Note: Replace "rsh" with "remsh" on HP-UX systems.

rsh $dbnode -c "startdb.sh $vollist"
```

Script to complete, recover and start a replica Oracle database

Use this script to complete, recover and start a replica Oracle database.

```
#!/bin/ksh
#
# script: startdb.sh <list_of_database_volumes>
#
# Sample script to complete, recover and start replica Oracle database.
#
# It is assumed that you have already performed the following
# steps:
# 1. Create the local volumes, file systems, and mount points for the
#     redo and archived logs, and then mount them.
# 2. Based on the text control file for the production database,
#     write a SQL script that creates a control file for the replica
#     database.
# 3. Create an initialization file for the replica database and place
#     this in the replica database's $ORACLE_HOME/dbs directory.
# 4. Copy the Oracle password file for the production database to the
#     replica database's $ORACLE_HOME/dbs directory.

export ORACLE_SID=REP1
export ORACLE_HOME=/rep/oracle/816
```

```
export PATH=$ORACLE_HOME/bin:$PATH
snapvoldg=snapdbdg
rep_mnt_point=/rep

# Import the snapshot volume disk group.

vxdg import $snapvoldg

# Mount the snapshot volumes (the mount points must already exist).

for i in $*
do
    fsck -V vxfs /dev/vx/rdisk/$snapvoldg/snap_$i
    mount -V vxfs /dev/vx/dsk/$snapvoldg/snap_$i ${rep_mnt_point}/${i}
done

# Fix any symbolic links required by the database.

cd ${rep_mnt_point}/dbase_vol
for i in 1 2 3 4 5 6 # adjust as required
do
    rm -f ./log$i
    ln -s ${rep_mnt_point}/dbase_logs/log$i ./log$i
done

# Remove the existing control file.

rm -f ${rep_mnt_point}/dbase_vol/cntrl1

# Create a new control file, recover and start the replica database.

svrmgrl <<!
connect internal
@c_file_create.sql
set autorecovery on
recover database until cancel using backup controlfile;
alter database open resetlogs;
quit
!
```

Script to start a replica Sybase ASE database

Use this script to start a replica Sybase ASE database.

```
#!/bin/ksh
#
# script: startdb.sh <list_of_database_volumes>
#
# Sample script to recover and start replica Sybase ASE database.

# Import the snapshot volume disk group.

vxdg import $snapvoldg

# Mount the snapshot volumes (the mount points must already exist).

for i in $*
do
    fsck -V vxfs /dev/vx/rdisk/$snapvoldg/snap_$i
    mount -V vxfs /dev/vx/dsk/$snapvoldg/snap_$i ${rep_mnt_point}/$i
done

# Start the replica database.
# Specify the -q option if you specified the "for external dump"
# clause when you quiesced the primary database.
# See the Sybase ASE 12.5 documentation for more information.

/sybase/ASE-12_5/bin/dataserver \
[-q] \
-sdatabase_name \
-d /sybevm/master \
-e /sybase/ASE-12_5/install/dbasename.log \
-M /sybase

# Online the database. Load the transaction log dump and
# specify "for standby_access" if you used the -q option
# with the dataserver command.

isql -Usa -Ppassword -SFMR <<!
[load transaction from dump_device with standby_access
go]
online database database_name [for standby_access]
go
```

```
quit  
!
```

Preparing a replica Oracle database

This appendix includes the following topics:

- [About preparing a replica Oracle database](#)
- [Text control file for original production database](#)
- [SQL script to create a control file](#)
- [Initialization file for original production database](#)
- [Initialization file for replica Oracle database](#)

About preparing a replica Oracle database

This appendix describes how to set up a replica off-host Oracle database to be used for decision support.

To prepare a replica Oracle database on a host other than the primary host

- 1 If not already present, install the Oracle software onto the host's local disks. The location of the Oracle home directory (\$ORACLE_HOME) is used for the database instance that is created from the snapshot volumes.

Note: In the examples shown here, the home directory is `/rep/oracle` in the local disk group, `localdg`. If required, you could instead choose to use the same file paths and database name as on the primary host.

- 2 In the local disk group, `localdg`, use the following command to create the volumes that are to be used for the redo logs and archived logs of the replicated database:

```
# vxassist -g diskgroup make volume size
```

For example, to create a 1-gigabyte redo log volume `rep_dbase_logs` and a 2-gigabyte archived log volume `rep_dbase_arch`:

```
# vxassist -g localdg make rep_dbase_logs 1g
# vxassist -g localdg make rep_dbase_arch 2g
```

- 3 Make the file systems for the redo logs and archive logs in the volumes created in the previous step using the following command:

```
# mkfs -V vxfs /dev/vx/rdisk/diskgroup/volume
```

In this example, the commands would be:

```
# mkfs -V vxfs /dev/vx/rdisk/localdg/rep_dbase_logs
# mkfs -V vxfs /dev/vx/rdisk/localdg/rep_dbase_arch
```

- 4 Create the mount points that are to be used to mount the new database. For example, create `/rep/dbase_vol` for the snapshot of the tablespace volume, `/rep/dbase_logs` for the redo logs, and `/rep/dbase_arch` for the archived logs:

```
# mkdir -p /rep/dbase_vol
# mkdir -p /rep/dbase_logs
# mkdir -p /rep/dbase_arch
```

- 5** Mount the redo log and archive log volumes on their respective mount points using the following command:

```
# mount -V vxfs /dev/vx/dsk/diskgroup/volume mount_point
```

In this example, the commands would be:

```
# mount -V vxfs /dev/vx/dsk/localdg/rep_dbase_logs \
  /rep/dbase_logs
# mount -V vxfs /dev/vx/dsk/localdg/rep_dbase_arch \
  /rep/dbase_arch
```

- 6** As the Oracle database administrator on the primary host, obtain an ASCII version of the current Oracle control file using the following SQL command:

```
alter database backup controlfile to trace;
```

This command writes a text version of the control file to the directory \$ORACLE_HOME/admin/dbase/udump.

See [“Text control file for original production database”](#) on page 524.

- 7** Modify the text version of the control file created in the previous step as described below to create a new SQL script to set up the replica database:

- If required, change the locations defined under LOGFILE for the log files. For example, change lines of the form:

```
GROUP N '/dbase_vol/logN' SIZE 52428288,
```

so that they read:

```
GROUP N '/rep/dbase_vol/logN' SIZE 52428288,
```

- If required, change the locations defined under DATAFILE for the tablespaces. For example, change lines of the form:

```
 '/dbase_vol/table',
```

so that they read:

```
 '/rep/dbase_vol/table',
```

- If required, change the following line:

```
CREATE CONTROLFILE REUSE DATABASE "odb" NORESETLOGS \
  ARCHIVELOG
```

so that it reads:

```
CREATE CONTROLFILE SET DATABASE "ndb" RESETLOGS \
NOARCHIVELOG
```

where *odb* is the name of the original database and *ndb* is the name of the replica database (DBASE and REP1 in the example). Note that to reduce unnecessary overhead, the new database is not run in archive log mode. See “[SQL script to create a control file](#)” on page 526.

- 8 Copy the Oracle initialization file for the original database to a new initialization file for the replica database.

For example, the Oracle initialization file for the original database is shown as `initdbase.ora`.

See “[Initialization file for original production database](#)” on page 526.

For example, the initialization file for the replica database is shown as `initREPl.ora`.

See “[Initialization file for replica Oracle database](#)” on page 528.

Edit the copied file and change the definitions of the following parameters:

`background_dump_dest` Background dump location.

`core_dump_dest` Core dump location.

`db_name` Database name to the name of the replica database.

`log_archive_dest` Archive log location, set equal to the path created in step 4 (for example, `/rep/dbase_arch`).

`log_archive_start` Archive log mode, `log_archive_start`, to FALSE.

`user_dump_dest` User dump location.

You may also wish to reduce the resource usage of the new database by adjusting the values of parameters such as `db_block_buffers`.

See the *Oracle Database Administrator's Guide* for more information.

- 9 Copy the Oracle remote password file (for example, `orapwdbase`) in `ORACLE_HOME/dbs` to a new file (for example, `orapwREPl`).

Text control file for original production database

The following example shows the text control file for the original production database.

```
/oracle/816/admin/dbase/udump/dbase_ora_20480.trc
Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
```

```

With the Partitioning option
JServer Release 8.1.6.0.0 - Production
ORACLE_HOME = /oracle/816
System name:      SunOS
Node name:        node01
Release:          5.8
Version:          Generic_108528-02
Machine:          sun4u
Instance name:    dbase
Redo thread mounted by this instance: 1
Oracle process number: 8
Unix process pid: 20480, image: oracle@node01
*** SESSION ID:(#.##) YYYY-MM-DD hh:mm:ss.sss
*** YYYY-MM-DD hh:mm:ss.sss
# The following commands will create a new control file and use it
# to open the database.
# Data used by the recovery manager will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "DBASE" NORESETLOGS ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 2
    MAXDATAFILES 70
    MAXINSTANCES 1
    MAXLOGHISTORY 226
LOGFILE
    GROUP 1 '/dbase_vol/log1'  SIZE 52428288,
# .
# . List of log files
# .
    GROUP N '/dbase_vol/logN'  SIZE 52428288
DATAFILE
    '/dbase_vol/ts1',
# .
# . List of tablespace datafiles
# .
    '/dbase_vol/tsN'
CHARACTER SET US7ASCII;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# All logs need archiving and a log switch is needed.

```

```
ALTER SYSTEM ARCHIVE LOG ALL;
# Database can now be opened normally.
ALTER DATABASE OPEN;
# No tempfile entries found to add.
#
```

SQL script to create a control file

The following example shows the SQL script to create a control file .

```
STARTUP NOMOUNT
CREATE CONTROLFILE SET DATABASE "REP1" RESETLOGS NOARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 2
    MAXDATAFILES 70
    MAXINSTANCES 1
    MAXLOGHISTORY 226
LOGFILE
    GROUP 1 '/rep/dbase_vol/log1' SIZE 52428288,
    # .
    # . List of log files
    # .
    GROUP N '/rep/dbase_vol/logN' SIZE 52428288
DATAFILE
    '/rep/dbase_vol/ts1',
    # .
    # . List of tablespace datafiles
    # .
    '/rep/dbase_vol/tsN'
CHARACTER SET US7ASCII
;
```

Initialization file for original production database

The following example shows the initialization file for the original production database.

```
#####+
# FILENAME          initdbase.ora
# DESCRIPTION       Oracle parameter file for primary database, dbase.
#####
db_block_size      = 8192
```

```

parallel_max_servers          = 30
recovery_parallelism         = 20
# db_writers                  = 25
# use_async_io                = TRUE
# async_io                    = 1
control_files                 = (/dbase_vol/cntrl1)
sort_area_size                = 15728640
parallel_max_servers         = 10
recovery_parallelism         = 4
compatible                    = 8.1.5
db_name                       = dbase
db_files                      = 200
db_file_multiblock_read_count = 32
db_block_buffers              = 30720 # 8k * 30720 approx 250MB
dml_locks                     = 500
hash_join_enabled             = FALSE

# Uncommenting the line below will cause automatic archiving if
# archiving has been enabled using ALTER DATABASE ARCHIVELOG.
log_archive_dest              = /archlog
log_archive_format             = dbase%t_%s.dbf
log_archive_start              = TRUE
# log_checkpoint_interval      = 1000000000

log_checkpoint_timeout         = 300
log_checkpoints_to_alert      = TRUE
log_buffer                     = 1048576
max_rollback_segments         = 220
processes                     = 300
sessions                       = 400
open_cursors                   = 200
transactions                   = 400
distributed_transactions       = 0
transactions_per_rollback_segment = 1
rollback_segments              =
(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s2
4,s25,s26,s27,s28,s29,s30)
shared_pool_size               = 7000000
cursor_space_for_time          = TRUE
audit_trail                    = FALSE
cursor_space_for_time          = TRUE
background_dump_dest           = /oracle/816/admin/dbase/bdump

```

```
core_dump_dest          = /oracle/816/admin/dbase/cdump
user_dump_dest         = /oracle/816/admin/dbase/udump
```

Initialization file for replica Oracle database

The following example shows the initialization file for the replica Oracle database.

```
##=====+
# FILENAME initREPl.ora
# DESCRIPTION Oracle parameter file for replica database, REPl.
#=====

db_block_size = 8192
parallel_max_servers          = 30
recovery_parallelism         = 20
# db_writers                  = 25
# use_async_io                = TRUE
# async_io                    = 1
control_files                 = (/rep/dbase_vol/cntrl1)
sort_area_size               = 15728640
parallel_max_servers          = 10
recovery_parallelism         = 4
compatible                   = 8.1.5
db_name                       = REPl
db_files                      = 200
db_file_multiblock_read_count = 32
db_block_buffers             = 10240
dml_locks                    = 500
hash_join_enabled            = FALSE
log_archive_start            = FALSE
log_archive_dest              = /rep/archlog
log_archive_format            = dbase%t_%s.dbf
log_checkpoint_timeout        = 300
log_checkpoints_to_alert     = TRUE
log_buffer                    = 1048576
max_rollback_segments        = 220
processes                     = 300
sessions                      = 400
open_cursors                  = 200
transactions                  = 400
distributed_transactions      = 0
transactions_per_rollback_segment = 1
rollback_segments            =
```

```
(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30)
shared_pool_size           = 7000000
cursor_space_for_time     = TRUE
audit_trail                = FALSE
cursor_space_for_time     = TRUE
background_dump_dest      = /rep/oracle/816/admin/REP1/bdump
core_dump_dest            = /rep/oracle/816/admin/REP1/cdump
user_dump_dest            = /rep/oracle/816/admin/REP1/udump
```


Glossary

address-length pair	Identifies the starting block address and the length of an extent (in file system or logical blocks).
asynchronous I/O	A format of I/O that performs non-blocking reads and writes. This enables the system to handle multiple I/O requests simultaneously.
atomic operation	An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.
block map	A file system is divided into fixed-size blocks when it is created. As data is written to a file, unused blocks are allocated in ranges of blocks, called extents. The extents are listed or pointed to from the inode. The term used for the data that represents how to translate an offset in a file to a file system block is the “block map” for the file.
boot disk	A disk used for booting an operating system.
buffered I/O	A mode of I/O operation (where I/O is any operation, program, or device that transfers data to or from a computer) that first transfers data into the Operating System buffer cache.
cache	Any memory used to reduce the time required to respond to an I/O request. The read cache holds data in anticipation that it will be requested by a client. The write cache holds data written until it can be safely stored on non-volatile storage media.
Cached Quick I/O	Cached Quick I/O allows databases to make more efficient use of large system memory while still maintaining the performance benefits of Quick I/O. Cached Quick I/O provides an efficient, selective buffering mechanism to complement asynchronous I/O.
cluster	A set of hosts that share a set of disks.
cluster-shareable disk group	A disk group in which the disks are shared between more than one host.
cold backup	The process of backing up a database that is not in active use.
command launcher	A graphical user interface (GUI) window that displays a list of tasks that can be performed by Veritas Volume Manager or other objects. Each task is listed with the object type, task (action), and a description of the task. A task is launched by

clicking on the task in the Command Launcher. concatenation A Veritas Volume Manager layout style characterized by subdisks that are arranged sequentially and contiguously.

concurrent I/O	A form of Direct I/O that does not require file-level write locks when writing to a file. Concurrent I/O allows the relational database management system (RDBMS) to write to a given file concurrently.
configuration database	A set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes). A single copy of a configuration database is called a configuration copy.
container	A physical storage device that can be identified by a directory name, a device name, or a file name.
copy-on-write	<p>A technique for preserving the original of some data. As data is modified by a write operation, the original copy of data is copied.</p> <p>Applicable to Storage Checkpoint technology, where original data, at the time of the Storage Checkpoint, must be copied from the file system to the Storage Checkpoint when it is to be overwritten. This preserves the frozen image of the file system in the Storage Checkpoint.</p>
database	A database is a collection of information that is organized in a structured fashion. Two examples of databases are Relational Databases (such as Oracle, Sybase, or DB2), where data is stored in tables and generally accessed by one or more keys and Flat File Databases, where data is not generally broken up into tables and relationships. Databases generally provide tools and/or interfaces to retrieve data.
dataserver	A logical concept of a Sybase instance. A Sybase instance contains databases and daemon processes that manage the data. A Sybase dataserver manages Sybase system databases and user created databases. Each Sybase dataserver is uniquely named when it is created.
DSS (Decision Support Systems)	Computer-based systems used to model, identify, and solve problems, and make decisions.
defragmentation	The act of reorganizing data to reduce fragmentation. Data in file systems become fragmented over time.
device file	A block- or character-special file located in the /dev directory representing a device.
device name	<p>The device name or address used to access a physical disk, such as <code>hdisk3</code>, which indicates the whole of disk 3.</p> <p>In a SAN environment, it is more convenient to use enclosure-based naming, which forms the device name by concatenating the name of the enclosure (such as <code>enc0</code>) with the disk's number within the enclosure, separated by an underscore</p>

	(for example, <code>enc0_2</code>). The term disk access name can also be used to refer to a device name.
direct I/O	An unbuffered form of I/O that bypasses the kernel's buffering of data. With direct I/O, data is transferred directly between the disk and the user application.
Dirty Region Logging	The procedure by which the Veritas Volume Manager monitors and logs modifications to a plex. A bitmap of changed regions is kept in an associated subdisk called a log subdisk.
disk access name	An alternative term for a device name.
disk array	A collection of disks logically and physically arranged into an object. Arrays provide benefits including data redundancy and improved performance.
disk cache	A section of RAM that provides a cache between the disk and the application. Disk cache enables the computer to operate faster. Because retrieving data from hard disk can be slow, a disk caching program helps solve this problem by placing recently accessed data in the disk cache. Next time that data is needed, it may already be available in the disk cache; otherwise a time-consuming operation to the hard disk is necessary.
disk group	A collection of disks that share a common configuration. A disk group configuration is a set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes) and their relationships. Each disk group has an administrator-assigned name and an internally defined unique ID. The root disk group (<code>rootdg</code>) is a special private disk group.
disk name	A Veritas Volume Manager logical or administrative name chosen for the disk, such as <code>disk03</code> . The term disk media name is also used to refer to the disk name.
Dynamic Multipathing	See DMP.
Decision Support Systems	See DSS.
DMP (Dynamic Multipathing)	A Veritas Volume Manager feature that allows the use of multiple paths to the same storage device for load balancing and redundancy.
error handling	Routines in a program that respond to errors. The measurement of quality in error handling is based on how the system informs the user of such conditions and what alternatives it provides for dealing with them.
evacuate	Moving subdisks from the source disks to target disks.
exabyte	A measure of memory or storage. An exabyte is approximately 1,000,000,000,000,000 bytes (technically 2 to the 60th power, or 1,152,921,504,606,846,976 bytes). Also EB.

extent	A logical database attribute that defines a group of contiguous file system data blocks that are treated as a unit. An extent is defined by a starting block and a length.
extent attributes	An extent allocation policy that is associated with a file and/or file system. See also address-length pair.
failover	The act of moving a service from a failure state back to a running/available state. Services are generally applications running on machines and failover is the process of restarting these applications on a second system when the first has suffered a failure.
file system	A collection of files organized together into a structure. File systems are based on a hierarchical structure consisting of directories and files.
file system block	The fundamental minimum size of allocation in a file system.
fileset	A collection of files within a file system.
fixed extent size	An extent attribute associated with overriding the default allocation policy of the file system.
fragmentation	Storage of data in non-contiguous areas on disk. As files are updated, new data is stored in available free space, which may not be contiguous. Fragmented files cause extra read/write head movement, slowing disk accesses.
gigabyte	A measure of memory or storage. A gigabyte is approximately 1,000,000,000 bytes (technically, 2 to the 30th power, or 1,073,741,824 bytes). Also GB, Gbyte, and G-byte.
HA (high availability)	The ability of a system to perform its function continuously (without significant interruption) for a significantly longer period of time than the combined reliabilities of its individual components. High availability is most often achieved through failure tolerance and inclusion of redundancy; from redundant disk to systems, networks, and entire sites.
hot backup	The process of backing up a database that is online and in active use.
hot pluggable	To pull a component out of a system and plug in a new one while the power is still on and the unit is still operating. Redundant systems can be designed to swap disk drives, circuit boards, power supplies, CPUs, or virtually anything else that is duplexed within the computer. Also known as hot swappable.
inode list	An inode is an on-disk data structure in the file system that defines everything about the file, except its name. Inodes contain information such as user and group ownership, access mode (permissions), access time, file size, file type, and the block map for the data contents of the file. Each inode is identified by a unique inode number in the file system where it resides. The inode number is used to find the inode in the inode list for the file system. The inode list is a series of inodes. There is one inode in the list for every file in the file system.

intent logging	A logging scheme that records pending changes to a file system structure. These changes are recorded in an intent log.
interrupt key	A way to end or break out of any operation and return to the system prompt by pressing Ctrl-C.
kilobyte	A measure of memory or storage. A kilobyte is approximately a thousand bytes (technically, 2 to the 10th power, or 1,024 bytes). Also KB, Kbyte, kbyte, and K-byte.
large file	A file more than two gigabytes in size. An operating system that uses a 32-bit signed integer to address file contents will not support large files; however, the Version 4 disk layout feature of VxFS supports file sizes of up to two terabytes.
large file system	A file system more than two gigabytes in size. VxFS, in conjunction with VxVM, supports large file systems.
latency	The amount of time it takes for a given piece of work to be completed. For file systems, this typically refers to the amount of time it takes a given file system operation to return to the user. Also commonly used to describe disk seek times.
load balancing	The tuning of a computer system, network tuning, or disk subsystem in order to more evenly distribute the data and/or processing across available resources. For example, in clustering, load balancing might distribute the incoming transactions evenly to all servers, or it might redirect them to the next available server.
load sharing	The division of a task among several components without any attempt to equalize each component's share of the load. When several components are load sharing, it is possible for some of the shared components to be operating at full capacity and limiting performance, while others components are under utilized.
LUN (logical unit number)	A method of expanding the number of SCSI devices that can be placed on one SCSI bus. Logical Unit Numbers address up to seven devices at each SCSI ID on an 8-bit bus or up to 15 devices at each ID on a 16-bit bus.
logical volume	See volume.
logical unit number	See LUN.
master node	A computer which controls another computer or a peripheral.
megabyte	A measure of memory or storage. A megabyte is approximately 1,000,000 bytes (technically, 2 to the 20th power, or 1,048,576 bytes). Also MB, Mbyte, mbyte, and K-byte.
metadata	Data that describes other data. Data dictionaries and repositories are examples of metadata. The term may also refer to any file or database that holds information about another database's structure, attributes, processing, or changes.
mirror	A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms mirror and plex can be used synonymously.

mirroring	A layout technique that mirrors the contents of a volume onto multiple plexes. Each plex duplicates the data stored on the volume, but the plexes themselves may have different layouts.
mount point	The directory path name at which a file system attaches to the file system hierarchy.
multithreaded	Having multiple concurrent or pseudo-concurrent execution sequences. Used to describe processes in computer systems. Multithreaded processes are one means by which I/O request-intensive applications can use independent access to volumes and disk arrays to increase I/O performance.
NBU	See Veritas NetBackup (NBU).
node	One of the hosts in a cluster.
object (VxVM)	An entity that is defined to and recognized internally by the Veritas Volume Manager. The VxVM objects include volumes, plexes, subdisks, disks, and disk groups. There are two types of VxVM disk objects—one for the physical aspect of the disk and the other for the logical aspect of the disk.
Online Transaction Processing	See OLTP.
online administration	An administrative feature that allows configuration changes without system or database down time.
OLTP (Online Transaction Processing)	A type of system designed to support transaction-oriented applications. OLTP systems are designed to respond immediately to user requests and each request is considered to be a single transaction. Requests can involve adding, retrieving, updating or removing data.
paging	The transfer of program segments (pages) into and out of memory. Although paging is the primary mechanism for virtual memory, excessive paging is not desirable.
parity	A calculated value that can be used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on data. The resulting parity is then written to the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and the parity.
partition	The logical areas into which a disk is divided.
persistence	Information or state that will survive a system reboot or crash.
petabyte	A measure of memory or storage. A petabyte is approximately 1,000 terabytes (technically, 2 to the 50th power).

plex	A duplicate copy of a volume and its data (in the form of an ordered collection of subdisks). Each plex is one copy of a volume with which the plex is associated. The terms mirror and plex can be used synonymously.
preallocation	Prespecifying space for a file so that disk blocks will physically be part of a file before they are needed. Enabling an application to preallocate space for a file guarantees that a specified amount of space will be available for that file, even if the file system is otherwise out of space.
Quick I/O	Quick I/O presents a regular Veritas File System file to an application as a raw character device. This allows Quick I/O files to take advantage of asynchronous I/O and direct I/O to and from the disk device, as well as bypassing the UNIX single-writer lock behavior for most file system files.
Quick I/O file	A regular UNIX file that is accessed using the Quick I/O naming extension (::cdev.vxfs:).
RAID	A Redundant Array of Independent Disks (RAID) is a disk array set up with part of the combined storage capacity used for storing duplicate information about the data stored in that array. This makes it possible to regenerate the data if a disk failure occurs.
repository	<p>A repository holds the name, type, range of values, source, and authorization for access for each data element in a database. The database maintains a repository for administrative and reporting use.</p> <p>Pertinent information, needed to display configuration information and interact with the database, is stored in the repository.</p>
root disk	The disk containing the root file system.
root disk group	A special private disk group on the system. The root disk group is named rootdg. However, starting with the 4.1 release of Veritas Volume Manager, the root disk group is no longer needed.
root file system	The initial file system mounted as part of the UNIX kernel startup sequence.
script	A file, containing one or more commands that can be run to perform processing.
shared disk group	A disk group in which the disks are shared by multiple hosts (also referred to as a cluster-shareable disk group).
sector	<p>A minimal unit of the disk partitioning. The size of a sector can vary between systems.</p> <p>A sector is commonly 512 bytes.</p>
segment	Any partition, reserved area, partial component, or piece of a larger structure.
System Global Area	See SGA.
single threading	The processing of one transaction to completion before starting the next.

slave node	A node that is not designated as a master node.
snapped file system	A file system whose exact image has been used to create a snapshot file system.
snapped volume	A volume whose exact image has been used to create a snapshot volume.
snapshot	A point-in-time image of a volume or file system that can be used as a backup.
snapshot file system	An exact copy of a mounted file system, at a specific point in time, that is used for online backup. A snapshot file system is not persistent and it will not survive a crash or reboot of the system.
snapshot volume	An exact copy of a volume, at a specific point in time. The snapshot is created based on disk mirroring and is used for online backup purposes.
spanning	A layout technique that permits a volume (and its file system or database) too large to fit on a single disk to distribute its data across multiple disks or volumes.
Storage Checkpoint	An efficient snapshot technology for creating a point-in-time image of a currently mounted VxFS file system. A Storage Checkpoint presents a consistent, point-in-time view of the file system by identifying and maintaining modified file system blocks.
storage class	Set of volumes with the same volume tag.
Storage Rollback	On-disk restore capability for faster recovery from logical errors, such as accidentally deleting a file. Because each Storage Checkpoint is a point-in-time image of a file system, Storage Rollback simply restores or rolls back a file or entire file system to a Storage Checkpoint.
stripe	A set of stripe units that occupy the same positions across a series of columns in a multi-disk layout.
stripe unit	Equally sized areas that are allocated alternately on the subdisks (within columns) of each striped plex. In an array, this is a set of logically contiguous blocks that exist on each disk before allocations are made from the next disk in the array.
stripe unit size	The size of each stripe unit. The default stripe unit size for VxVM is 32 sectors (16K). For RAID 0 striping, the stripe unit size is 128 sectors (64K). For VxVM RAID 5, the stripe unit size is 32 sectors (16K). A stripe unit size has also historically been referred to as a stripe width.
striping	A layout technique that spreads data across several physical disks using stripes. The data is allocated alternately to the stripes within the subdisks of each plex.
subdisk	A consecutive set of contiguous disk blocks that form a logical disk segment. Subdisks can be associated with plexes to form volumes.
superuser	A user with unlimited access privileges who can perform any and all operations on a computer. In UNIX, this user may also be referred to as the “root” user. On Windows/NT, it is the “Administrator.”

tablespace	A tablespace is a storage structure (containing tables, indexes, large objects, and long data) that allows you to assign the location of database and table data directly onto containers. Tablespaces reside in database partition groups.
terabyte	A measure of memory or storage. A terabyte is approximately 1,000,000,000,000 bytes (technically, 2 to the 40th power, or 1,000 GB). Also TB.
throughput	A measure of work accomplished in a given amount of time. For file systems, this typically refers to the number of I/O operations in a given period of time.
unbuffered I/O	I/O that bypasses the file system cache for the purpose of increasing I/O performance (also known as direct I/O).
Veritas Enterprise Administrator	Application that is required to access graphical user interface (GUI) functionality.
NBU (Veritas NetBackup)	A product that lets you back up, archive, and restore files, directories, or raw partitions that reside on your client system.
VVR (Veritas Volume Replicator)	A feature of Veritas Volume Manager, VVR is a data replication tool designed to contribute to an effective disaster recovery plan.
Version Checkpoint	In a DB2 UDB EEE or ESE Data Partitioning Feature (DPF) environment, a Version Checkpoint is a set of checkpoints that spans multiple partitions. A Version Checkpoint contains one Storage Checkpoint per partition. A Storage Checkpoint is an efficient snapshot technology for creating a point-in-time image of a currently mounted VxFS file system. A Storage Checkpoint presents a consistent, point-in-time view of the file system by identifying and maintaining modified file system blocks.
volume	A logical disk device that appears to applications, databases, and file systems as a physical disk partition. A logical disk can encompass multiple or one to many physical volumes.
volume layout	A variety of layouts that allows you to configure your database to meet performance and availability requirements. This includes spanning, striping (RAID-0), mirroring (RAID-1), mirrored stripe volumes (RAID-0+1), striped mirror volumes (RAID-1+0), and RAID 5.
volume manager objects	Volumes and their virtual components. See object (VxVM).
VVR	See “Veritas Volume Replicator (VVR).”
vxfs or VxFS	The acronym for Veritas File System.
vxvm or VxVM	The acronym for Veritas Volume Manager.

Index

Symbols

\$DSQUERY 64
\$LD_LIBRARY_PATH 64
\$PATH 64
\$SYBASE 64
/etc/default/vxassist defaults file 467
/etc/default/vxcdsconvert defaults file 467
/etc/default/vxdg defaults file 467
/etc/default/vxdisk defaults file 468
/etc/default/vxencap defaults file 468
/etc/vx/darecs file 462

A

absolute path names
 using with Quick I/O 46, 65
absolute pathnames
 use with symbolic links 44, 63
access type 455
accessing
 Quick I/O files with symbolic links 44, 63
activation
 default 478
ACTIVE state 206
AIX coexistence label 455
alignment 457
 changing 472
allocating file space 41, 60
analyze 411
analyzing I/O statistics 83, 94
ARCHIVELOG mode 196
archiving
 using NetBackup 301
ATTACHING state 206
attribute
 CDS 478
attributes
 autogrow 215, 219
 autogrowby 153, 215
 cache 219
 cachesize 219
 dcolen 255

attributes (*continued*)
 highwatermark 153, 215
 init 151
 maxautogrow 215
 mirdg 227
 mirvol 227
 ncachemirror 219
 ndcomirror 151, 255
 ndcomirs 214
 newvol 225
 nmirror 151, 225
 regionsize 153
 snapvol 221, 227
 source 221, 227
 syncing 212, 239
 tuning the autogrow feature 153
auto disk type 455
autogrow
 tuning 241
autogrow attribute 215, 219
autogrow feature
 tuning attributes 153
autogrowby attribute 153, 215
automatic backups 301

B

backing up
 using NetBackup 301
backup
 of cluster file systems 167
 of online databases 155
backups
 created using snapshots 212
 creating for volumes 200
 creating using instant snapshots 212
 creating using third-mirror snapshots 244
 for multiple volumes 228, 249
BCV 138
benefits of Concurrent I/O 102
benefits of Quick I/O 35
block size 453

- blockdev --rereadpt 481
- blockmap for a snapshot file system 263
- break-off snapshots
 - emulation of 205
- BROKEN state 206
- Business Continuance Volume (BCV) 138

C

- cache
 - autogrow attributes 153
 - creating for use by space-optimized snapshots 151
 - for space-optimized instant snapshots 135
- cache advisory
 - checking setting for 87, 99
- cache attribute 219
- cache hit ratio
 - calculating 83, 95
- cache objects
 - creating 215
 - enabling 216
 - listing snapshots in 241
- Cached Quick I/O
 - customizing 85, 96
 - determining files to use 82, 94
 - disabling individual files 85, 97
 - enabling individual files 85, 97
 - making settings persistent 85, 97
 - prerequisite for enabling 78, 90
- caches
 - creating 215
 - deleting 244
 - finding out snapshots configured on 244
 - growing 243
 - listing snapshots in 241
 - removing 244
 - resizing 243
 - shrinking 243
 - stopping 244
 - used by space-optimized instant snapshots 204
- cachesize attribute 219
- calculating cache hit ratio 83, 95
- cascade instant snapshots 207
- cascaded snapshot hierarchies
 - creating 232
- CDS
 - attribute 478
 - changing setting 473
 - creating DGs 462

- CDS (*continued*)
 - creating disks 461
 - disk group alignment 454
 - disk group device quotas 456
 - disks 454
- CDS disk groups
 - alignment 479
 - joining 473
 - moving 473
 - setting alignment 472
- CDS disks
 - creating 460
- changing CDS setting 473
- changing default CDS setting 473
- changing disk format 469
- changing file sizes 41, 60
- chgrp command 42, 58
- chmod command
 - commands
 - chmod 78
- chown command 42, 58
 - commands
 - chown 78
- cluster file systems
 - off-host backup of 167
- co-existence label 455
- coexistence
 - VxVM and LVM disks 395
- collecting I/O statistics 82, 94
- commands
 - chgrp 42, 58
 - chown 42, 58
 - fsadm command 52, 73
 - grep 80, 92
 - ls 50, 71
 - mount 40, 58
 - qio_convertdbfiles 44, 48, 64, 68
 - qio_getdbfiles 44, 47, 64, 67
 - qioadmin 84, 96
 - qiomkfile 51–53, 72–73
 - qiostat 82, 94
 - setext 42, 58
 - vxedit 420
 - vxtunefs 86, 98
- concepts 451
- Concurrent I/O
 - benefits 102
 - disabling 105–106
 - enabling 102, 105

- configuration
 - LVM 395
- configuration VxVM 395
- conversion
 - errors 503
 - speed 413
 - vxconvert 397
- converting
 - Quick I/O files back to regular files Quick I/O
 - converting back to regular files 65
 - Quick I/O files back to regular files Quick I/O
 - converting back to regular files 47
 - regular files to Quick I/O files 48, 68
- converting a data Storage Checkpoint to a nodata Storage Checkpoint 278
- converting non-CDS disks to CDS 463
- converting non-CDS disks to CDS disks 463
- copy-on-write
 - used by instant snapshots 203
- copy-on-write technique 267, 271
- CREADs 84, 95
- creating
 - Quick I/O files 41, 61
 - symbolic links to access Quick I/O files 41, 60
- creating a DRL log 475
- creating a multi-volume support file system 326
- creating CDS disk groups 462
- creating CDS disks 460–461
- creating DRL logs 475
- creating non-CDS disk groups 473
- creating pre-version 110 disk groups 473
- cross-platform data migration 23
- cross-platform data sharing
 - recovery file 490
- current-rev disk groups 457
- customizing Cached Quick I/O 85, 96

D

- data Storage Checkpoints definition 272
- database
 - specifying type for Quick I/O 46–47, 65
- database performance
 - using Quick I/O 35
- databases
 - incomplete media recovery 197
 - integrity of data in 135
 - online backup of 155
 - preparing off-host replica for Oracle 521
 - rolling back 196

- databases (*continued*)
 - using Storage Checkpoints 195
- DCO
 - adding to volumes 143
 - adding version 0 DCOs to volumes 253
 - considerations for disk layout 147
 - dissociating version 0 DCOs from volumes 257
 - effect on disk group split and join 147
 - moving log plexes 146, 256
 - reattaching version 0 DCOs to volumes 257
 - removing version 0 DCOs from volumes 257
 - specifying storage for version 0 plexes 256
- dcolen attribute 255
- deactivate
 - disk group 427
 - volume group 427
- decision support
 - using point-in-time copy solutions 177
- default activation 478
- default CDS setting
 - changing 473
- defaults files 463, 467
- deport
 - disk group 427
- destroy
 - disk group 427
- determining
 - if Quick I/O installed and enabled 71
- device nodes
 - controlling access for volume sets 321
 - displaying access for volume sets 321
 - enabling access for volume sets 320
 - for volume sets 319
- device quotas 456, 478
 - displaying 478
 - setting 474
- disable
 - mirror 435
- disabled file system
 - snapshot 261
- disabling Cached Quick I/O for a file 85, 97
- disabling Concurrent I/O 105–106
- disabling qio_cache_enable flag 79, 90
- disabling Quick I/O 56, 75
- disk
 - access type 455
 - change format 469
 - designate 434
 - disk space 398

- disk (*continued*)
 - evacuate 435
 - labels 469
 - LVM 469
 - offline 434
 - online 435
 - recover 435
 - replace 433
 - replacing 474
 - disk access 453
 - disk format 454
 - disk group 425
 - rename 433
 - disk group alignment 472
 - displaying 479
 - Disk Group Split/Join 136
 - disk groups 389, 455
 - alignment 457
 - creating 473
 - joining 473
 - layout of DCO plexes 147
 - non-CDS 457
 - upgrading 474
 - disk headers 397
 - disk quotas
 - setting 474
 - disk types 454
 - disks 389
 - coexistence 395
 - effects of formatting or partitioning 480
 - layout of DCO plexes 147
 - display
 - disk group 428
 - DMP 435
 - logical volume 428
 - physical volume 428
 - volume 428
 - VxVM volumes 428
 - displaying device quotas 478
 - displaying disk group alignment 479
 - displaying DRL log size 478
 - displaying DRL map size 478
 - displaying log map values 479
 - displaying log size 478
 - displaying v_logmap values 479
 - displaying volume log map values 479
 - DMP
 - display 435
 - DRL log size
 - displaying 478
 - setting 475
 - DRL logs
 - creating 475
 - DRL map length 476
 - DRL map size
 - displaying 478
 - setting 475
 - dsync flag 64
 - Dynamic Multipathing 424
- ## E
- EMC Symmetrix 138
 - EMC TimeFinder 138
 - enabling
 - Quick I/O 39, 57
 - enabling Cached Quick I/O for a file 85, 97
 - enabling Concurrent I/O 102, 105
 - enabling qio_cache_enable flag 79, 90
 - encapsulating volumes 324
 - encapsulation 469
 - ENOSPC 286
 - equivalent command 420
 - error messages 503
 - Example
 - analyze LVM groups 410
 - conversion 410
 - failed conversion 410
 - list 410
 - list disk information 410
 - list LVM volume group information 410
 - listvg 410
 - LVM to VxVM 410
 - vxprint output 410
 - example
 - Failed Analysis 410
 - export
 - volume group 427
 - extending a file 41, 60
 - extending Quick I/O files 51, 72
 - extracting file list for Quick I/O conversion 47, 67
- ## F
- FastResync
 - Persistent 134
 - snapshot enhancements 201

features

- task monitor 390

file

- space allocation 41, 60

file fragmentation

- reporting on 65

file system 426**file systems**

- growing to accommodate Quick I/O files 51, 72
- mounting for shared access 168

fileset

- primary 269

FileSnaps

- about 293

- data mining, reporting, and testing 299

- virtual desktops 298

- write intensive applications 298

- backup 296

- best practices 298

- block map fragmentation 296

- concurrent I/O 295

- copy-on-write 295

- creating 295

- properties 294

- reading from 296

- using 297

FlashSnap 133**FlashSnap Agent 138****freezing and thawing, relation to Storage**

- Checkpoints 269

fsadm command 52, 73**fsat 262****fscdsadm 484****fscdsconv 489****fsck 278****fsckptadm**

- Storage Checkpoint administration 274

fsvoladm 326**full backups 301****full-sized instant snapshots 135, 202**

- creating 221

- creating volumes for use as 217

fullinst snapshot type 239**G****grep command 80, 92****growing**

- file systems 51, 72

- Quick I/O files 51, 72

H

- highwatermark attribute 153, 215

Hot Relocation 424

- how to access a Storage Checkpoint 276

- how to create a Storage Checkpoint 275

- how to mount a Storage Checkpoint 276

- how to remove a Storage Checkpoint 275

- how to unmount a Storage Checkpoint 278

I

- I/O block size 453

- ID block 455

import

- disk group 426–427

- volume group 426

improving

- database performance 35

- incremental backups 301

- init attribute 151

instant snapshots

- backing up multiple volumes 228

- cascaded 207

- creating backups 212

- creating for volume sets 229

- creating full-sized 221

- creating space-optimized 218

- creating volumes for use as full-sized 217

- displaying information about 237

- dissociating 236

- full-sized 135, 202

- improving performance of synchronization 240

- reattaching 175, 193, 233

- refreshing 233

- removing 236

- restoring volumes using 235

- space-optimized 135, 204

- splitting hierarchies 237

- synchronizing 239

intent log

- multi-volume support 324

- intent logging 200

J**join**

- subdisk 435

- joining CDS disk groups 473

- joining disk groups 473

- L**
- length listing 480
 - licensing 466
 - link objects 206
 - linked break-off snapshots 206
 - creating 226
 - linked third-mirror snapshots
 - reattaching 234
 - list file for Quick I/O conversion 47, 67
 - list LVM 410
 - listing disk groups 480
 - listing disks 480
 - listing offset and length information 473
 - log size
 - displaying 478
 - setting 475
 - Logical Volume 392–394
 - logical volume
 - convert 431
 - split 431
 - synchronize 431
 - Logical Volume Manager 389
 - ls command 50, 71
 - LUNs
 - thin provisioning 114
 - lvchange 420
 - lvcreate 420
 - lvextend 421
 - LVM 389
 - metadata 398
 - LVM disks 469
 - LVM names
 - symbolic names 404
 - LVM VGRA 397
 - lvsync 421
- M**
- mapping
 - LVM device nodes 404
 - VxVM device nodes 404
 - master device path 65
 - maxautogrow attribute 153, 215
 - messages
 - error 503
 - metadata
 - multi-volume support 324
 - migrating to thin storage 114
 - minor device numbers 457
 - mirbrk snapshot type 239
 - mirdg attribute 227
 - mirror
 - disable 435
 - disks 389
 - logical volume 430
 - plex 430
 - remove 435
 - repair 435
 - mirroring 424
 - mirroring and striping 424
 - mirrors
 - creating snapshot 246
 - mirvol attribute 227
 - mirvol snapshot type 239
 - mkqio.dat file 47–49, 67–68, 75
 - mkqio.sh script options
 - report on file fragmentation 65
 - mount
 - mounting a Storage Checkpoint 276
 - pseudo device 277
 - mount command 40, 58
 - mounting
 - shared-access file systems 168
 - mounting a Storage Checkpoint 278
 - mounting a Storage Checkpoint of a cluster file system 278
 - move
 - subdisk 435
 - moving CDS disk groups 473
 - moving disk group objects 473
 - Multi-Volume Support 315
 - multi-volume support 323
 - creating a MVS file system 326
- N**
- name space
 - preserved by Storage Checkpoints 268
 - names
 - defining for snapshot volumes 249
 - ncachemirror attribute 219
 - ndcomirror attribute 151, 255
 - ndcomirs attribute 214
 - NetBackup
 - overview 301
 - newvol attribute 225
 - nmirror attribute 151, 224–225
 - nodata Storage Checkpoints 278
 - nodata Storage Checkpoints definition 273

O

- objects
 - moving 473
- off-host backup of cluster file systems
 - using point-in-time copy solutions 167
- offset
 - listing 480
- offset information 480
- online database backup
 - using point-in-time copy solutions 155
- Online Migration 424
- operating system data 453
- Oracle Disk Manager 35
 - benefits 36

P

- performance
 - improving for instant snapshot synchronization 240
 - snapshot file systems 262
- persistence
 - for Cached Quick I/O settings 85, 97
- Persistent FastResync 134
- physical volumes 392–394
- platform block 455
- plex attribute 225
- plexes
 - adding to snapshots 251
 - converting to snapshot 248
 - moving 146, 256
- point-in-time copy solutions
 - applications 138
 - for decision support 177
 - for off-host cluster file system backup 167
 - for online database backup 155
- PREADs 84, 95
- preallocating space for Quick I/O files 42, 58
- primary fileset relation to Storage Checkpoints 269
- private region 454
- pseudo device 277
- Public Region 394
- public region 454
- pvchange 422
- pvdisplay 421

Q

- qio_cache_enable flag
 - disabling 79, 90

- qio_cache_enable flag (*continued*)
 - enabling 79, 90
- qio_convertdbfiles command 44, 48, 64, 68
- qio_getdbfiles command 44, 47, 64, 67
- qioadmin command 84, 96
- qiomkfile command 51–53, 72–73
 - options for creating files
 - symbolic links 41, 60
- qiostat
 - output of 83, 94
- qiostat command 82, 94
- Quick I/O
 - accessing regular VxFS files as 43, 62
 - benefits 35
 - converting files to 48, 68
 - determining file fragmentation before
 - converting 65
 - determining status 71
 - disabling 56, 75
 - enabling 39, 57
 - environment variable requirements 64
 - extending files 51, 72
 - extracting file list for conversion 47, 67
 - improving database performance with 35
 - list file for conversion 47, 67
 - preallocating space for files 42, 58
 - setting environment variables for 66
 - showing resolution to a raw device 51, 72
 - specifying database name for 64
 - using relative and absolute pathnames 44, 63
- quotas
 - hard limit 292

R

- RAID-5 424
- raw device nodes
 - controlling access for volume sets 321
 - displaying access for volume sets 321
 - enabling access for volume sets 320
 - for volume sets 319
- read-only Storage Checkpoints 276
- recovery
 - using Storage Checkpoints 195
- recovery file, cross-platform data sharing 490
- reduce
 - volume group 428
- regionsize attribute 153, 214–215
- relative pathnames
 - use with symbolic links 44, 63

- removable Storage Checkpoints definition 274
- remove
 - disk 429
 - volume 429
 - volume group 427, 429
- removing
 - non-VxFS files from mkqio.dat file 48
- removing non-VxFS files from mkqio.dat file 46, 68
- rename
 - disk group 433
- repair
 - mirror 435
- replacing disks 474
- resetlogs option 198
- resizing a file 41, 60
- restore
 - disk group 436
- restoring
 - using NetBackup 301
- restoring CDS disk labels 469
- restoring disk labels 469
- resyncfromoriginal snapback 211
- resyncfromreplica snapback 211
- resynchronize
 - volumes 432
- resynchronizing
 - snapshots 136
- rootability
 - root disk 398
 - root volume 398
- RUN_SERVER file 65

S

- sa_password 65
- SAM
 - vgdisplay 402
- setext command 42, 58
- setting CDS disk group alignment 472
- setting device quotas 474
- setting disk quotas 474
- setting DRL log size 475
- setting DRL map length 476
- setting DRL map size 475
- setting environment variables for Quick I/O 66
- setting log size 475
- settings
 - making Cached Quick I/O persistent 79, 91
- shared access
 - mounting file systems for 168

- showing
 - Quick I/O file resolved to raw device 51, 72
- SmartMove feature 114
- SmartSync Recovery Accelerator 23
- SmartTier 23, 315
 - multi-volume support 324
- SMIT 437
- snap volume naming 211
- snapabort 201
- snapback
 - defined 201
 - merging snapshot volumes 250
 - resyncfromoriginal 211
 - resyncfromreplica 211, 250
- snapclear
 - creating independent volumes 252
- snapmir snapshot type 239
- snapof 264
- snapped file systems 259
 - performance 262
 - unmounting 260
- snapread 262
- snapshot file systems 259
 - blockmap 263
 - creating 264
 - data block area 263
 - disabled 261
 - fscat 262
 - fuser 260
 - mounting 264
 - multiple 260
 - on cluster file systems 260
 - performance 262
 - read 262
 - super-block 263
- snapshot hierarchies
 - creating 232
 - splitting 237
- snapshot mirrors
 - adding to volumes 231
 - removing from volumes 232
- snapshots
 - adding mirrors to volumes 231
 - adding plexes to 251
 - backing up multiple volumes 228, 249
 - backing up volumes online using 212
 - cascaded 207
 - converting plexes to 248
 - creating a hierarchy of 232

- snapshots (*continued*)
 - creating backups using third-mirror 244
 - creating for volume sets 229
 - creating full-sized instant 221
 - creating independent volumes 252
 - creating instant 212
 - creating linked break-off 226
 - creating snapshots of 208
 - creating space-optimized instant 218
 - creating third-mirror break-off 223
 - creating volumes for use as full-sized instant 217
 - defining names for 249
 - displaying information about 252
 - displaying information about instant 237
 - dissociating instant 236
 - emulation of third-mirror 205
 - finding out those configured on a cache 244
 - full-sized instant 202
 - hierarchy of 207
 - improving performance of synchronization 240
 - instant 135
 - linked break-off 206
 - listing for a cache 241
 - merging with original volumes 250
 - on multiple volumes 211
 - preparing volumes 141
 - reattaching instant 175, 193, 233
 - reattaching linked third-mirror 234
 - refreshing instant 233
 - removing 248
 - removing instant 236
 - removing linked snapshots from volumes 232
 - removing mirrors from volumes 232
 - restoring from instant 235
 - resynchronization on snapback 211
 - resynchronizing 136
 - resynchronizing volumes from 250
 - space-optimized instant 204
 - synchronizing instant 239
 - third-mirror 135
 - use of copy-on-write mechanism 203
- snapsize 264
- snapstart 201
- snapvol attribute 221, 227
- snapwait 224, 227
- source attribute 221, 227
- space-optimized instant snapshots 135, 204
 - creating 218
- spaceopt snapshot type 239
- sparse files 49
- specifying
 - master device path 65
- specifying database name for Quick I/O 64
- split
 - subdisk 435
- states
 - of link objects 206
- storage cache 135
 - used by space-optimized instant snapshots 204
- Storage Checkpoints 137
 - accessing 276
 - administration of 274
 - converting a data Storage Checkpoint to a nodata Storage Checkpoint with multiple Storage Checkpoints 281
 - creating 196, 275
 - data Storage Checkpoints 272
 - database recovery 195
 - definition of 268
 - difference between a data Storage Checkpoint and a nodata Storage Checkpoint 279
 - freezing and thawing a file system 269
 - mounting 276
 - multi-volume support 324
 - nodata Storage Checkpoints 273, 278
 - operation failures 286
 - pseudo device 277
 - read-only Storage Checkpoints 276
 - removable Storage Checkpoints 274
 - removing 275
 - space management 286
 - synchronous vs. asynchronous conversion 279
 - types of 272
 - unmounting 278
 - using the fsck command 278
 - writable Storage Checkpoints 276
- Storage Rollback
 - implementing using Storage Checkpoints 195
 - using VxDBA 196
- subdisk
 - join 435
 - move 435
 - split 435
- super-block 263
- Sybase
 - environment variables for Quick I/O 64

symbolic links

- advantages and disadvantages 43, 62
- to access Quick I/O files 44, 63

Symmetrix 138

synchronization

- controlling for instant snapshots 239
- improving performance of 240

syncing attribute 212, 239

syncpause 240

syncresume 240

syncstart 240

syncstop 240

syncwait 240

T

task monitor 390

thin provisioning

- using 114

Thin Reclamation 120

thin storage

- using 114

third-mirror

- snapshots 135

third-mirror break-off snapshots

- creating 223

third-mirror snapshots 205

TimeFinder 138

tool

- vxconvert 397

tools

- vxconvert 395
- vxdiskadm 395

troubleshoot

- errors 503

tunefstab file

- adding tuning parameters to 80, 91

tuning parameters

- adding to tunefstab file 80, 91

U

unattended backups 301

unmount 278

- a snapped file system 260

upgrading disk groups 474

upgrading pre-version 110 disk groups 474

utilities. *See* commands**V**

v_logmap

- displaying 479

verifying caching using vxfstune parameters 80, 92

verifying vxtunefs system parameters 80, 92

Veritas Cached Quick I/O 23

Veritas Extension for Oracle Disk Manager 23

Veritas Quick I/O 23

Veritas Volume Manager 389

vgchange 422

vgcreate 422

vgdisplay 422

vgexport 423

vgextend 422

vgimport 424

VGRA 394

vgreduce 423

vgremove 423

vgscan 423

vgsync 423

volbrk snapshot type 239

volume

- concatenated 428
- logical 428
- RAID-5 428
- striped 428

Volume Manager

- features 390

volume sets

- adding volumes to 316
- administering 315
- controlling access to raw device nodes 321
- creating 316
- creating instant snapshots of 229
- displaying access to raw device nodes 321
- enabling access to raw device nodes 320
- listing details of 317
- raw device nodes 319
- removing volumes from 317
- starting 318
- stopping 318

volumes 389

- adding DCOs to 143
- adding snapshot mirrors to 231
- adding to volume sets 316
- adding version 0 DCOs to 253
- backing up 200
- backing up online using snapshots 212

- volumes (*continued*)
 - creating for use as full-sized instant snapshots 217
 - creating from snapshots 252
 - creating snapshots 247
 - displaying information about snapshots 252
 - dissociating version 0 DCOs from 257
 - Logical Volume 392–394
 - merging snapshots 250
 - naming snap 211
 - physical volumes 392–394
 - preparing for full-sized instant snapshots 150
 - preparing for instant snapshots 141
 - reattaching version 0 DCOs to 257
 - removing linked snapshots from 232
 - removing snapshot mirrors from 232
 - removing version 0 DCOs from 257
 - restoring from instant snapshots 235
 - resynchronizing from snapshots 250
 - specifying storage for version 0 DCO plexes 256
 - taking multiple snapshots 211
- VX_SNAPREAD 262
- vxassist 420–421
 - adding DCOs to volumes 255
 - creating cache volumes 215
 - creating snapshots 244
 - creating volumes for use as full-sized instant snapshots 218
 - displaying information about snapshots 252
 - dissociating snapshots from volumes 252
 - merging snapshots with volumes 250
 - moving DCO log plexes 146
 - moving DCO plexes 256
 - removing version 0 DCOs from volumes 257
 - resynchronizing volumes from snapshots 250
 - snapabort 201
 - snapback 201
 - snapshot 201
 - snapstart 201
 - specifying storage for version 0 DCO plexes 256
 - taking snapshots of multiple volumes 249
- vxcache
 - listing snapshots in a cache 241
 - resizing caches 243
 - starting cache objects 216
 - stopping a cache 244
 - tuning cache autogrow 242
- vxcached
 - tuning 241
- vxcached daemon 153
- vxcdsconvert 463
- vxconvert 411
- vxdc
 - dissociating version 0 DCOs from volumes 257
 - reattaching version 0 DCOs to volumes 257
 - removing version 0 DCOs from volumes 257
- vxctl enable 481
- vxdg 422
 - vxdg init 462
 - vxdg split 472
- vxdisk 421–422
- vxdisk scandisks 481
- vxdisk set 422
- vxdiskadd 422
- vxdiskadm 461, 463
- vxdisksetup 460
- vxedit 420, 422
 - removing a cache 244
 - removing instant snapshots 236
 - removing snapshots from a cache 244
- VxFS 426
- vxinfo 423
- vxmake
 - creating cache objects 215
- vxmend 424
- vxplex 424
 - converting plexes to snapshots 248
- vxprint 422–423
 - displaying DCO information 147, 256
 - displaying snapshots configured on a cache 244
 - verifying if volumes are prepared for instant snapshots 214
- vxrecover 421
- vxresize 420
- vxsnap
 - adding snapshot mirrors to volumes 231
 - administering instant snapshots 203
 - backing up multiple volumes 228
 - controlling instant snapshot synchronization 240
 - creating a cascaded snapshot hierarchy 232
 - creating full-sized instant snapshots 221, 227
 - creating linked break-off snapshot volumes 227
 - creating space-optimized instant snapshots 219
 - displaying information about instant snapshots 237–238
 - dissociating instant snapshots 236

vxsnap (*continued*)

- preparing volumes for instant snapshot operations 143
 - preparing volumes for instant snapshots 214
 - reattaching instant snapshots 175, 193, 233
 - reattaching linked third-mirror snapshots 234
 - refreshing instant snapshots 233
 - removing a snapshot mirror from a volume 232
 - restore 203
 - restoring volumes 235
 - splitting snapshot hierarchies 237
- vxtunefs** command 86, 98
- commands
 - vxtunefs 80, 92
- VxVM** 389
- devices 453
 - features 390
 - metadata 398
- VxVM** names
- symbolic link 404
- VxVM** volumes
- resynchronize 432
- vxvol** 421, 424, 476
- vxvset**
- adding volumes to volume sets 316
 - controlling access to raw device nodes 321
 - creating volume sets 316
 - creating volume sets with raw device access 320
 - listing details of volume sets 317
 - removing volumes from volume sets 317
 - starting volume sets 318
 - stopping volume sets 318

W

- warning messages
- Specified region-size is larger than the limit on the system 213
- writable Storage Checkpoints 276